НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

УДК: 004.032.26

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___” _____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: «*Інформаційна система підтримки розпізнавання комп’ютерних гестур*»

Виконав: студент *4* курсу, групи *ІС-51*
Потильчак Даниіл Григорович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *доц., к.т.н., доц. Баклан І.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *ст. викладач Москаленко Н.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц. каф. ТК, к.т.н., доц. Ткач М.М.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент Потильчак Д.Г. _____
(підпис)

Київ – 2019 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з шести розділів, містить 18 рисунків, 11 таблиць, 1 додаток, 15 джерел.

Дипломний проект присвячений розробці інформаційної системи підтримки розпізнавання комп’ютерних гестур.

У розділі інформаційного забезпечення були вказані вхідні та вихідні дані до системи, було описано звідки формується навчаюча вибірка для згорткової нейронної мережі та описана структура зберігання даних, необхідних для роботи системи.

Розділ математичного забезпечення присвячений обґрунтуванню обраного підходу вирішення поставленої задачі та опису його роботи.

Програмне забезпечення описує засоби розробки системи, вимоги до технічного забезпечення та архітектуру програмного забезпечення.

У технологічному розділі описана інструкція користувача та проведене тестування комплексу задач.

КОМП’ЮТЕРНІ ГЕСТУРИ, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ.

ABSTRACT

Structure and scope of work. Diploma project consists of six sections, contains 18 drawings, 11 tables, 1 applications, 15 sources.

The diploma project is devoted to the development of information support system for recognizing computer gestures.

The information support section describes the input and output data to the system, describes where is the training set is and describes the storage structure necessary for the system to work.

The section of mathematical support is devoted to the substantiation of the chosen approach to the solution and the description of its work.

The software describes system development tools, hardware requirements and software architectures.

The technology section describes the user's manual and tests a set of tasks.

COMPUTER GESTURES, IMAGE CLASSIFICATION, MACHINE LEARNING, NEURAL NETWORKS, CONVOLUTIONAL NEURAL NETWORKS.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

БД – база даних.

ЗНМ – згорткова нейронна мережа.

МН – машинне навчання.

НМ – нейронна мережа.

ОС – операційна система

Гіперпараметри нейронної мережі – всі можливі змінні, які характеризують та формують нейронну мережу. Суть побудови гарної нейронної мережі полягає у підборі оптимальних гіперпараметрів (а також у правильному та обширному датасеті)

Градієнтний спуск – метод оптимізації, який використовується у машинному навчанні для приближення вагів нейронних мереж до оптимальних значень, з якими мережа зможе краще вирішувати поставлену задачу.

Датасет – набір даних (у випадку даного програмного забезпечення, такими даними є зображення гестур, створених користувачем) необхідних для тренування нейронних мереж.

Комп’ютерна гестура – жест, який людина може намалювати мишкою або рукою (у разі сенсорного екрану). Представляє собою зображення довільної кривої. Користувач на власний розсуд придумає форму гестури, керуючись своїми представленнями, асоціаціями тощо.

ВСТУП

Стрімкий розвиток обчислювальної техніки призвів до можливості широкого використання нейронних мереж, навчання яких потребує значних обчислювальних потужностей. У минулому, коли власне були створені перші моделі нейронних мереж, основані на роботі нейронів мозку людини, не було наявно достатньо швидких обчислювальних пристроїв, які б могли за адекватну кількість часу натренувати нейронну мережу [1]. Особливо складною є задача розпізнавання зображень, оскільки на вхід до нейронної мережі подається значна кількість вхідних даних – всі пікселі зображення з усіх кольорових каналів. В даний момент, маючи всі необхідні обчислювальні потужності, людство може застосовувати нейронні мережі для вирішення багатьох задач, які довгий час було неможливо вирішити іншими алгоритмами [2].

Розпізнавання жестів є дуже актуальною та важливою темою в області комп'ютерних наук. Її мета - інтерпретація людських жестів за допомогою математичних алгоритмів. Жести можуть виходити з будь-якого тілесного руху або стану, але зазвичай виникають з обличчя або руки. В даній роботі буде йти річ про жести, які людина може намалювати мишкою або рукою (у разі сенсорного екрану) на комп’ютері. Будемо називати такі жести комп’ютерними гестурами. Користувачі можуть використовувати прості гестури для керування та взаємодії з пристроями. Для інтерпретації мови жестів було використано багато підходів з використанням різних алгоритмів комп'ютерного зору. Розпізнавання жестів можна розглядати як спосіб, за допомогою якого комп'ютери починають розуміти мову людського тіла, створюючи тим самим більш насичений міст між машинами і людьми, ніж примітивні текстові інтерфейси користувача або навіть графічні інтерфейси користувача, які все ще обмежують більшість вхідних даних на клавіатурі і миші і взаємодіяти природно без будь-яких механічних пристроїв [3].

Отже, гестура являє собою зображення і тому розпізнавання гестур можна назвати задачею класифікації зображень. Хоч сучасні комп'ютери і мають змогу тренувати доволі складні нейронні мережі за адекватних час, але все ж таки у випадку, коли вхідними даними до нейронної мережі є зображення, кількість нейронів стає значно великою, що доволі сильно затрудняє навчання та використання таких мереж. Цю проблему вирішає нових підхід до розміщення нейронів між собою. Архітектура згорткових нейронних мереж частково вирішує проблему надвеликої кількості нейронів. Початкове зображення, проходячи через шари згортки та агрегування, зменшує свою розмірність, але при цьому зберігає свої головні ознаки та патерни, на основі яких нейронна мережа зможе досить точно класифікувати зображення [4].

Наведемо основні визначення, що є важливими для теми розпізнавання комп'ютерних гестур.

Комп'ютерна гестура – жест, який людина може намалювати мишкою або рукою (у разі сенсорного екрану). Представляє собою зображення довільної кривої. Користувач на власний розсуд придумає форму гестури, керуючись своїми представленнями, асоціаціями тощо.

Згорткова нейронна мережа - спеціальна архітектура штучних нейронних мереж, запропонована Яном Лекуном в 1988 році [5] і націлена на ефективне розпізнавання образів [6], входить до складу технологій глибокого навчання (англ. deep learning). Використовує деякі особливості зорової кори [7], в якій були відкриті так звані прості клітини, що реагують на прямі лінії під різними кутами, і складні клітини, реакція яких пов'язана з активацією певного набору простих клітин [8].

Аугментація даних - це метод, який можна використовувати для штучного розширення розміру навчаючої виборки для нейронних мереж, створюючи модифіковані версії даних. Оскільки вхідними даними у нашому випадку є зображення, то аугментація зображень - це створення

модифікованих версії зображень. Підготовка глибоких моделей нейронних мереж на більшій кількості даних може призвести до більш вправних моделей, а техніки збільшення можуть створити варіації зображень, які можуть покращити здатність моделей підходити до узагальнення того, що вони вивчили до нових образів [9].

Дипломний проект присвячений розробці інформаційної системи підтримки розпізнавання комп'ютерних гестур.

Практичне значення одержаних результатів. Розробка інформаційної системи підтримки розпізнавання комп’ютерних систем для полегшення взаємодії користувача з обчислювальним пристроєм.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Для початку розглянемо ідеологію жестів. Існуючі дослідження в сфері психології свідчать, що до 40% інформації надходить від жестів та міміки. Багато різних жестів люди використовують у повсякденному житті. Доволі цікаво, що в одному й тому ж відділу мозку виконується розпізнавання мови і письмових символів. Вчені це вияснили доволі давно, оскільки обом способам спілкування між людьми притаманні певні правилам граматики та існує однаковий словниковий запас. Жести є досить значною частиною нашого повсякденного спілкування, вони є всюди. Люди використовують їх, щоб виразити певні емоції, або передати певну інформацію, доповнити мовлення, додати до своїх слів емоційне забарвлення тощо.

Нажаль, такий значущий спосіб комунікації людини з навколишнім середовищем досить обмежено використовується під час сучасної взаємодії з різними інформаційними системами. До недавнього часу це пояснювалося відсутністю можливості адекватно зчитувати жести людини, але на даний момент ми маємо всі необхідні технології для того, щоб впровадити використання жестів в нашу модель взаємодії людини з різними обчислювальними пристроями.

Комп’ютерна гестура – це жест, намальований людиною мишкою, пальцем (у разі сенсорного екрану) або спеціальним пером на графічному планшеті. Сьогодні найбільшого поширення гестури здобули в управлінні смартфонів з сенсорним екраном. Але це досить примітивні функції для основної навігації в операційній системі смартфона. Якщо взяти більш складніші гестури, то за допомогою них, на прикладі зі смартфоном, можна не тільки об'єднати всі навігаційні кнопки, а просто змінити всю взаємодію користувача з операційною системою.

Суть використання гестур - уникнути посередника між користувачем і обчислювальним пристроєм (тобто уникнути кнопок). Продовжуючи приклад зі смартфоном: користувач використовує звичайну кнопку для переходу на робочій стіл, а з використанням гестур, він міг би без ніяких кнопок виконати цю дію, наприклад, провівши лінію вниз. Це також значно економить екранний простір, у випадку відсутності кнопок.

Якщо подивитися зі сторони психології, то саме гестури будуть набагато зрозуміліше ніж кнопки. Наприклад, згортаючи вікно за допомогою гестури маху вниз, мозок користувача очікує операції згортання вікна від системи, бо якщо представити вікно у вигляді фізичного об'єкта, то подібні дії призвели б до подібних наслідків, відповідно до фізики реальних об'єктів. Також, в деяких ситуаціях новому користувачу може бути досить складно розібратися з кнопками і за що вони відповідають, і одночасно з цим, використання гестур в даному випадку може дозволити користувачу інтуїтивно навчитися взаємодіяти з програмним продуктом та отримати почуття вільного і зрозумілого керування.

1.1.1 Опис процесу діяльності

Процес діяльності користувача полягає у створенні будь-якої кількості гестур та подальшому введенні гестур на розпізнавання, які будуть класифікуватися програмою як існуючі в системі гестури, або як нерозпізнані. Послідовність дій представлена на рисунку 1.1.

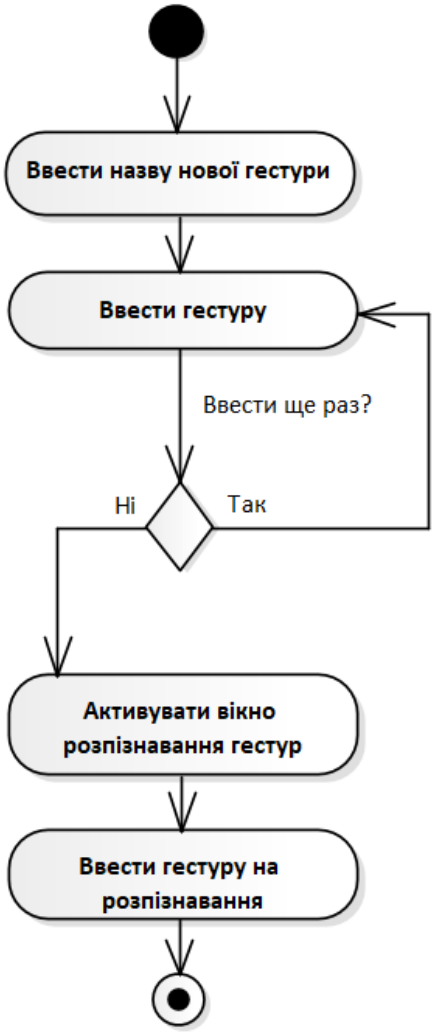


Рисунок 1.1 – Діаграма діяльності користувача

Спочатку користувач повинен ввести назву нової гестури. Потім він повинен придумати який саме вигляд буде мати гестура, яку він за допомогою комп'ютерної миші або свого пальця (у випадку сенсорного екрану) намалює у відповідній області. Далі користувач має змогу скільки завгодно повторювати введення цієї гестури, що буде створювати нові

приклади зображень гестур у відповідному каталозі. Це збільшить точність розпізнавання, особливо якщо вже є гестура, яка досить схожа на нову. Для подальшого використання програмного продукту, користувачу необхідно буде активувати зчитування гестур, вибравши відповідний пункт в меню програми, та ввести гестуру на розпізнавання.

1.1.2 Опис функціональної моделі

Специфікацію функціональної поведінки системи представлено у вигляді діаграми варіантів використання. Схему структурну варіантів використання наведено у графічному матеріалі.

На схемі наведено всі функції системи та описано акторів, які будуть їх використовувати. Із системою буде взаємодіяти тільки один актор: користувач – це особа, яка може працювати із основними функціями системи (створення, редагування, видалення, розпізнавання гестур).

При створенні гестури, користувач вводить назву та будь-яку кількість прикладів зображень гестур. Потім є можливість видалити будь-яку гестуру. Також користувач має можливість додати нові приклади гестур.

Відповідно до визначених варіантів використання виявлено функціональні вимоги та встановлено їх пріоритетність. Результат наведено в таблиці 1.1.

Таблиця 1.1 – Функціональні вимоги

Варіант використання	**Функціональна вимога**	**Пріоритет**
1. Створення гестури	Система надає можливість створити гестуру.	Високий
1.1. Введення назви гестури	Система надає можливість ввести назву гестури.	Високий
1.2. Введення зображення гестури	Система надає можливість ввести гестуру у вигляді довільного рисунка.	Високий
2. Редагування гестури	Система надає можливість редагувати існуючі гестури.	Середній

Продовження таблиці 1.1

2.1. Додавання прикладу гестури	Система надає можливість додати нові приклади зображення гестур до існуючої гестури.	Середній
2.2. Видалення гестури	Система надає можливість видалити існуючі гестури.	Високий
3. Розпізнавання гестури	Система надає можливість вводити гестури на розпізнавання.	Високий
3.1. Введення зображення гестури	Система надає можливість ввести гестуру та зчитує її.	Високий
3.2. Знаходження відповідності між введеною гестурою та існуючими	Система має порівняти введену гестуру з існуючими та встановити наявність або відсутність відповідності.	Високий

1.2 Огляд наявних аналогів

Нині існують програмні продукти, які здійснюють роботу з гестурами. В ході пошуку схожих рішень було виявлено деякі підходи зі схожою функціональністю. Найбільш поширені аналоги:

- gMote
- GestureSign
- StrokesPlus

Перелічені програмні продукти справно виконують свою основну функцію – створення користувачем своїх гестур та призначення їм певних дій, розпізнавання нових введених користувачем гестур та пошук відповідності між введеною та існуючими в системі гестурами. При аналізі функціональності даних продуктів було встановлено, що для зіставлення введеної гестури з існуючим, використовується дещо інший підхід: про введені гестури, лінія розбивається на точки, які потім формують кусочно-неперервну лінію або вручну задаються точки, з я яких формується лінія.

Знаходження відповідності з існуючими гестурами у даному випадку виконується за допомогою множини точок. З таким підходом програма не використовує велику кількість невидимих людському оку особливостей та характеристик ліній, таких як форма вигинів, опуклостей та увігнутостей, власні ледь помітні особливості почерку людини та багато інших ознак, які навіть неможливо сформулювати. Приклад аналогу можна побачити на рисунку 1.3.

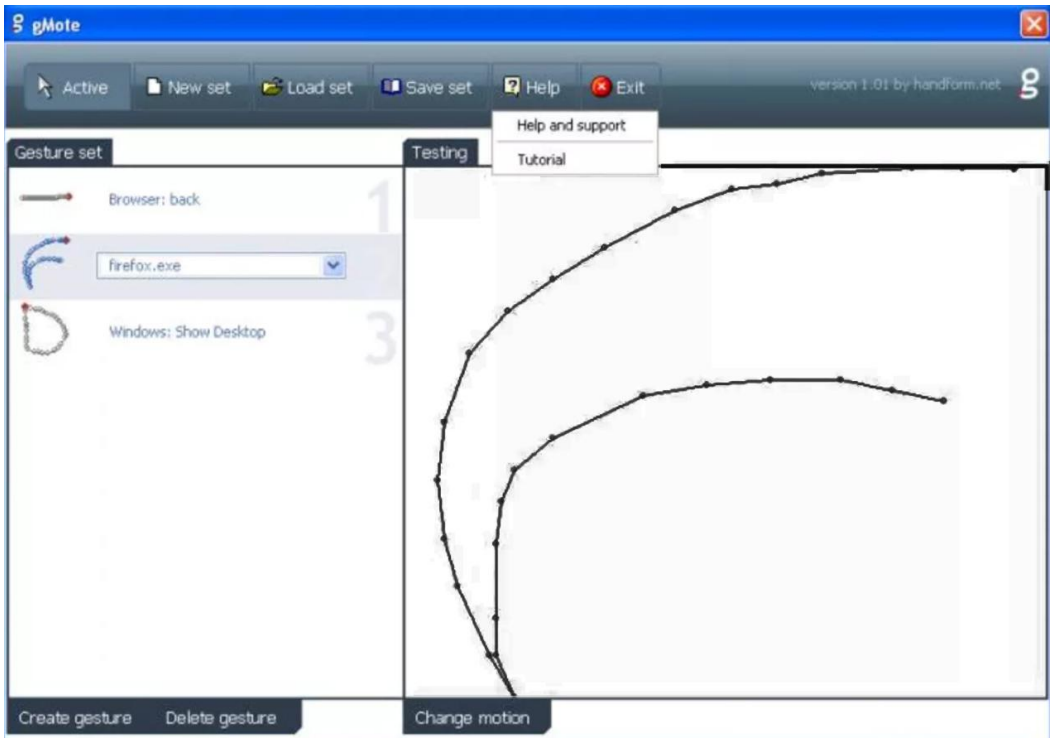


Рисунок 1.3 – Приклад аналогу

Особливістю даного програмного продукту є використання згорткової нейронної мережі, що дає можливість більш точно знаходити відповідності між введеними та існуючими гестурами. Також з таким підходом, при достатній кількості прикладів гестур, можливо використовувати дуже складні форми гестур і досить схожі гестури, але з незначними відмінностями, які згорткова нейронна мережа буде фіксувати.

1.3 Постановка задачі

1.3.1 Призначення розробки

Запропонований програмний продукт призначений для розпізнавання гестур, які вводить користувач. Це, в свою чергу, призведе до полегшення комунікації між користувачем та обчислювальним пристроєм. При використанні різних електронних пристроїв, ми нерідко виконуємо складні комплексні але часто повторювані дії, на виконання яких кожного разу витрачаємо деякий час. З даним програмним продуктом є можливість створити гестуру, яка буде розпізнаватися при подальшому введені гестур, що дає можливість використати цей функціонал в інших програмних продуктах для призначення гестурам будь-яких дій і простим введенням гестур запускати ці дії на виконання в необхідний момент. Це значно спростить взаємодію користувача з програмним продуктом. Наприклад, можна буде виконувати різні дії з головного вікна програми вводячи відповідні гестури, а не тратити час на багаточисельні переходи по меню.

1.3.2 Цілі та задачі розробки

Мета роботи – створення системи розпізнавання гестур для полегшення формулювання користувачем управляючих команд.

Дана система вирішує актуальну задачу розпізнавання введених користувачем гестур за допомогою згорткової нейронної мережі, яка має змогу знайти певні патерни в попередньо створених гестурах та на їх основі класифікувати нові гестури, що вводить користувач. Завдяки створеній системі у розробників різних програмних продуктів є інструмент для впровадження нового способу комунікації користувача з програмою. При інтеграції системи в різноманітні програмні продукти, є можливість значно спростити керування програмним продуктом, тим самим зменшивши час на

виконання певних управляючих команд або інших дій, що призведе до збільшення ефективності виконуваної користувачем роботи.

Для реалізації поставленої мети необхідно розв’язати наступні задачі:

- створити інтерфейс для використання програмного забезпечення, в якому користувач матиме можливість розпізнати гестуру, переглянути існуючі гестури, добавити до них нові приклади, видалити існуючі гестури, створити нову гестуру, вийти з програми;
- створити інтерфейс, за допомогою якого користувач матиме можливість намалювати гестуру (для створення нової гестури, додавання прикладу до існуючої гестури чи для розпізнавання введеної гестури);
- розробити функціонал для зменшення розмірності зображень з гестурами для покращення роботи основного алгоритму розпізнавання;
- розробити функціонал для аугментації датасету;
- створення архітектури згорткової нейронної мережі, яка найкраще підходить для вирішення поставленої задачі розпізнавання гестур;
- навчання згорткової нейронної мережі для розпізнавання гестур на створених зображеннях гестур;
- підбір гіперпараметрів згорткової нейронної мережі для найкращої точності розпізнавання гестур в умовах невеликої кількості прикладів зображень гестур.

Висновок до розділу

У даному розділі було розглянуто предметне середовище, було визначено та описано процес діяльності, що наведений у структурній схемі діяльності. Іншим результатом є створення функціональної моделі системи: опис користувачів системи, їх ролі та можливі дії у системі. Функціональна модель представлена у вигляді структурної схеми варіантів використання.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідними даними для даного програмного продукту є:

- зображення гестур у папці «gestures» в каталозі програмного продукту, які зберігаються у файлах з розширенням «PNG» та мають найменування у такому вигляді: «назва гестури»_«порядковий номер прикладу гестури». Ці зображення використовуються для тренування згорткової нейронної мережі на розпізнавання даних гестур. Також система використовує назви файлів з зображеннями гестур та перший файл зображення кожної гестури для формування вікна списку гестур.
- зображення гестури, введеного користувачем на розпізнавання. Система обробляє отримане зображення за допомогою натренованої на існуючих гестурах згорткової нейронної мережі та виводить результат.
- файл зі збереженими вагами натренованої на існуючих гестурах згорткової нейронної мережі, які зберігаються у файлі з назвою «weights.h5»

Головним інформаційним елементом даного програмного продукту є комп’ютерна гестура, яку вводить користувач. Гестура має вигляд рисунка (зазвичай одна або декілька нерозривних кривих або ліній), який створює користувач шляхом її малювання комп’ютерною мишкою або пальцем (у разі сенсорного екрану) чорною лінією на білому фоні розміром 500x500 пікселів у відповідному вікні програми. Форму гестури придумує сам користувач керуючись власними побажаннями, ідеями та асоціаціями. Чим сильніше відрізняються наявні в системі гестури, тим легше згортковій нейронній мережі буде відрізняти гестури одну від одної та правильно розпізнавати гестури.

2.2 Вихідні дані

Вихідними даними даного програмного продукту є назви гестур, які вводить користувач під час роботи програми, або повідомлення про те, що гестура, введена користувачем, не розпізнана.

Після того, як користувач намалює гестуру для розпізнавання, система обробляє отримане зображення за допомогою натренованої на існуючих гестурах згорткової нейронної мережі, яка в своєму шару виходу видає оцінки (числа) відповідності введеної гестури до кожної з існуючих гестур. З цих оцінок вибирається найбільша і система видає назву існуючої гестури, яка відповідає найбільшій оцінці. У випадку, коли згорткова нейронна мережа видала всі оцінки, які нижче за встановлений поріг, система повідомляє що гестуру не розпізнано.

Також система зберігає ваги натренованої на існуючих гестурах згорткової нейронної мережі у файлі з назвою «weights.h5». Це необхідно для того, щоб при подальшому використанні системи, вона могла при запуску не тренувати згорткову нейронну мережу з нуля, а використати вже існуючи ваги натренованої мережі з попереднього запуску програми, тим самим заощадивши обчислювальні ресурси комп'ютера та значно зменшивши час запуску програми.

2.3 Структура масивів інформації

При запуску програма автоматично зчитує наявні файлі в каталозі з гестурами та бере звідти назви існуючих гестур та малюнки гестур (самий перший екземпляр) для відображення цієї інформації у вікні списку існуючих гестур. Також отримані назви гестур програма сортує у встановленому порядку (за номером символів в системі кодування Unicode [10]). Це необхідно для того, щоб певні нейрони, які відповідають за відповідну гестуру, після наступних запусків програмного продукту з завантаженням

попередніх вагів натренованої згорткової мережі відповідали за одну і ту ж назву гестури. Інакше програма буде працювати неправильно.

Ваги натренованої на існуючих гестурах згорткової нейронної мережі з попереднього сеансу використання системи зберігаються у файлі з назвою «weights.h5».

Всі приклади гестур, які збережені в системі, зберігаються у каталозі «gestures» у вигляді зображень з розширенням «PNG» та мають найменування у такому вигляді: «назва гестури»_«порядковий номер прикладу гестури». Приклад каталогу «gestures» з наявними гестурами можна побачити на рисунку 2.1.

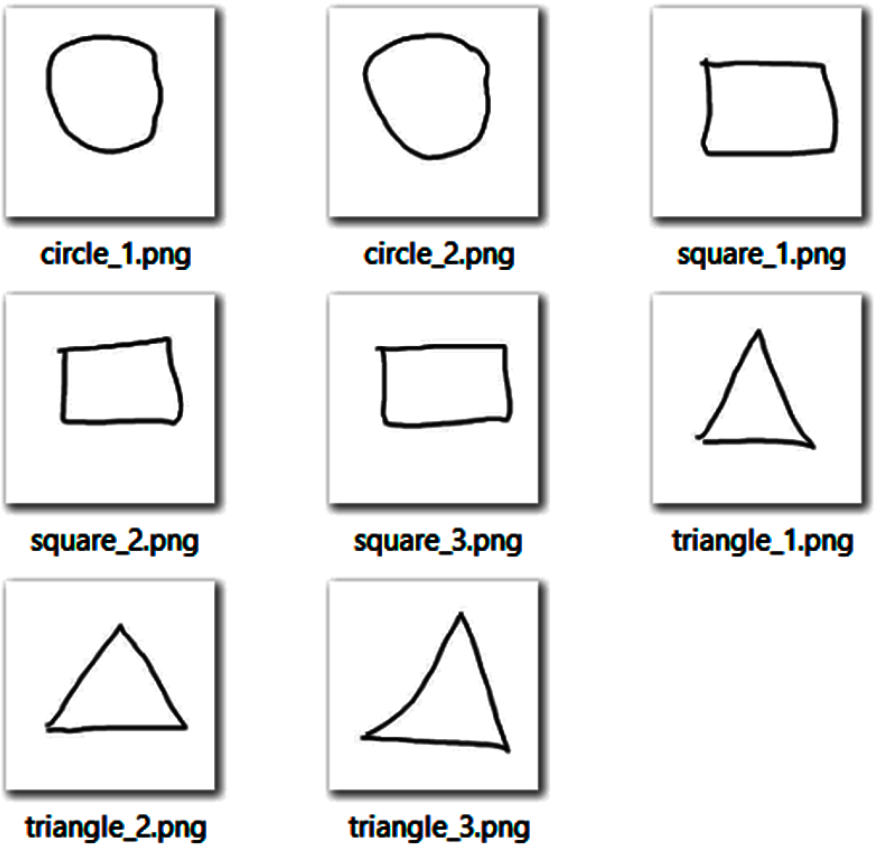


Рисунок 2.1 – Приклад каталогу «gestures»

Висновок до розділу

У розділі інформаційного забезпечення було описано вхідні дані, які необхідні для правильної роботи системи. Оскільки алгоритм розпізнавання гестур – це згорткова нейронна мережа, то їй необхідна деяка кількість зображень гестур для того, щоб навчитися ці самі гестури розпізнавати надалі. Чим більший розмір датасету – тим краще і точніше алгоритм зможе розпізнавати гестури. Якщо деякі гестури будуть сильно схожі одна на одну, то для достатньої точності розпізнавання потрібно буде ввести дещо більше прикладів відповідних гестур. У разі частих помилок, у користувача завжди є можливість добавити нові приклади зображень гестур.

Вихідними даними системи є результати роботи згорткової нейронної мережі над введеною користувачем гестурою на розпізнавання. Система виводить або назву введеної гестури, якщо вона її розпізнала, або повідомлення про те, що гестуру не розпізнано.

Також у розділі було описано структуру зберігання даних. База даних не використовується за непотрібністю, оскільки система не зберігає ніяких масивів даних. Майже всі дані (крім вагів згорткової нейронної мережі у файлі формату «h5») зберігаються у вигляді зображень з розширенням «PNG». Таким чином, у системи є всі необхідні методи для доступу до каталогу «gestures», де зберігаються зображення гестур, і використання бази даних призведе тільки до зменшення швидкості роботи системи, а також до надлишкових компонентів, що збільшить можливість виникнення проблем при використанні даного програмного продукту.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Існують гестури, які мають вигляд рисунка (зазвичай одна або декілька нерозривних кривих або ліній), які вводить користувач. Форму гестури створює користувач керуючись власними побажаннями, ідеями та асоціаціями. Кожна гестура має свою назву та будь-яку кількість прикладів рисунків того, як ця гестура має виглядати. Зазвичай у кожної гестури є 3-5 прикладів зображень. Це і буде нашою навчаючою вибіркою, на основі якої буде тренуватися згорткова нейронна мережа для того, щоб проаналізувати нову гестуру, введену користувачем та віднести її до вже існуючих гестур, або зробити висновок що гестуру не розпізнано. Також навчаюча вибірка буде розширення шляхом аугментації існуючих зображень з прикладами гестур.

3.2 Математична постановка задачі

Дано: зображення комп’ютерних гестур у вигляді файлів з розширенням «PNG».

Знайти: значення вагів згорткової нейронної мережі, з якими дана мережа зможе вдало розпізнавати гестури.

Саме навчання нейронних мереж полягає у тому, що під час ітераційного процесу вираховується функція втрат, на основі якої за допомогою різних методів (в нашому випадку – градієнтний спуск) корегуються ваги мережі. Складністю даної задачі є досить обмежений та маленький датасет зображень гестур. Для того, щоб згорткова нейронна мережа мала достатню точність розпізнавання, використовується доволі неглибока архітектура мережі та аугментація наявних зображень гестур для збільшення датасету зображень.

3.3 Обґрунтування методу розв’язання

Згорткові нейронні мережі дуже схожі зі звичайними нейронними мережами: вони складаються з нейронів, які мають певні ваги і зміщення. Кожен нейрон отримує деякі входи, виконує скалярний добуток і проходить через функцію активації. Вся мережа виражає єдину диференційовану функцію оцінки: від необроблених пікселів зображення з одного кінця до класових балів на іншому. І вони все ще мають функцію втрат.

Архітектура згорткової нейронної мережі (англ. ConvNet) робить явне припущення, що входи є зображеннями, що дозволяє кодувати певні властивості в архітектурі. Таким чином поширення вперед (англ. forward propagation) скрізь нейронну мережу більш ефективно для реалізації та значно зменшують кількість параметрів у мережі [11].

Звичайні нейронні мережі отримують вхід один вектор і перетворюють його через ряд прихованих шарів. Кожен прихований шар складається з набору нейронів, де кожен нейрон повністю з'єднаний з усіма нейронами попереднього шару, і де нейрони в одному шарі функціонують повністю незалежно і не мають жодних з'єднань між собою. Останній повністю пов'язаний шар називається «шаром виходу», а в класифікаційних параметрах він представляє бали (оцінки) класів.

Звичайні нейронні мережі не мають достатнього масштабу для повних зображень. У CIFAR-10 (датасет з рукописними цифрами від 0 до 9) зображення розміром лише 32x32x3 (32 ширина, 32 висота, 3 кольорових канали), тому один повністю пов'язаний нейрон у першому прихованому шарі звичайної нейронної мережі матиме 32 * 32 * 3 = 3072 ваг. Ця сума все ще здається керованою, але очевидно, що ця повністю пов'язана структура не масштабується до великих зображень. Наприклад, зображення більш респектабельного розміру, наприклад 500x500x3, призведе до нейронів, які мають 500 * 500 * 3 = 7500000 ваг. Більше того, ми майже напевно хочемо мати кілька таких нейронів, тому параметри швидко збільшуються.

Очевидно, що повне підключення всіх нейронів є марнотратним, і величезна кількість параметрів швидко призведе до перенасичення (англ. overfitting).

Згорткові нейронні мережі користуються тим фактом, що вхідні дані складаються з зображень і вони обмежують архітектуру більш розумним способом. Зокрема, на відміну від звичайної нейронної мережі, шари згорткової нейронної мережі мають нейрони, розташовані в 3-х вимірах: ширина, висота, глибина (тут глибина відноситься до третього виміру вхідних даних, а конкретно до кольорового каналу, а не до глибини повної нейронної мережі, яка може стосуватися загальної кількості шарів у мережі). Наприклад, вхідні зображення в CIFAR-10 є вхідним обсягом активацій, а обсяг має розміри 32x32x3 (ширина, висота, глибина відповідно). Як ми незабаром побачимо, нейрони в шарі будуть з'єднані лише з малим регіоном шару перед ним, а не з усіма нейронами повнозв'язним чином. Крім того, кінцевий вихідний шар для CIFAR-10 має розміри 1x1x10, оскільки до кінця архітектури згорткової нейронної мережі ми зменшимо повне зображення в єдиний вектор класових оцінок (балів), розташованих уздовж глибини. На рисунку 3.1 зображена архітектура звичайної повнозв'язної нейронної мережі.

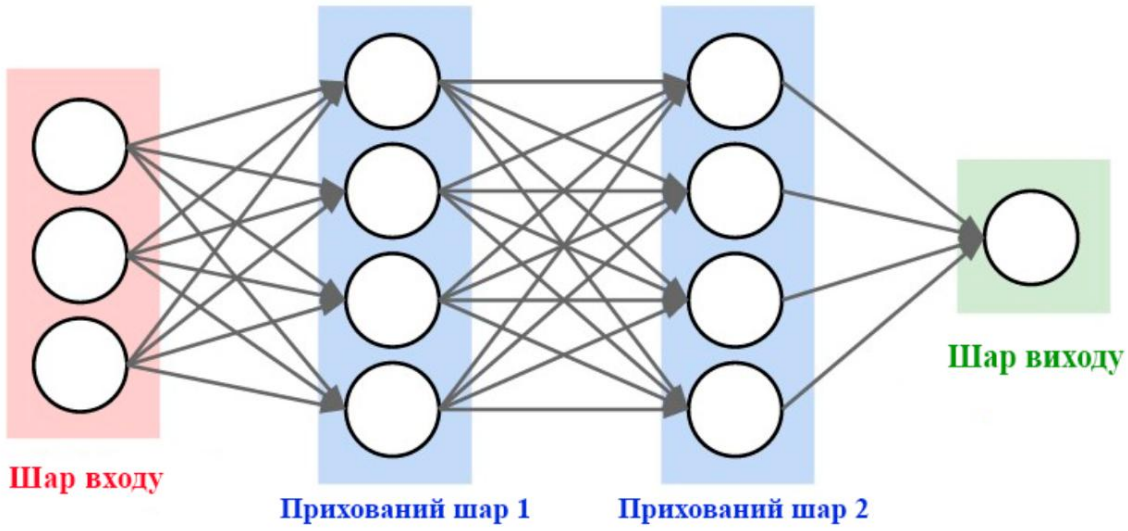


Рисунок 3.1 – Архітектура нейронної мережі

На рисунку 3.2 – спрощена архітектура згорткової нейронної мережі (англ. ConvNet), на якій можна побачити різницю у розташуванні нейронів у порівнянні зі звичайною нейронною мережею. Червоний вхідний шар є нашим зображенням, прихований шар має ширину і висоту яка відповідає розмірам зображення і глибину рівну кількості кольорових каналів (червоний, зелений, синій у звичайному зображенні).

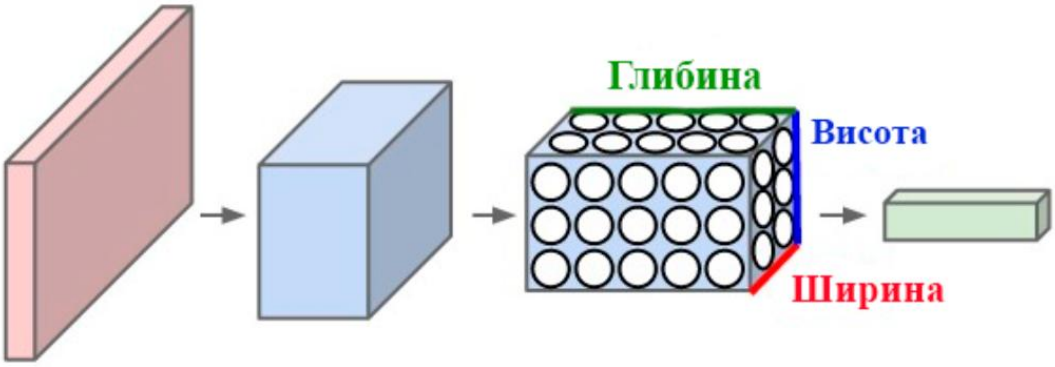


Рисунок 3.2 – Спрощена архітектура згорткової нейронної мережі

Згорткова нейронна мережа є послідовністю шарів, і кожен шар мережі перетворює один обсяг активацій нейронів в інший за допомогою диференційованої функції. Також ми використовуємо функції активації: softmax для шару виходу (вибирає найбільше значення оцінки) та RELU, графік якої можна побачити на рисунку 3.3.

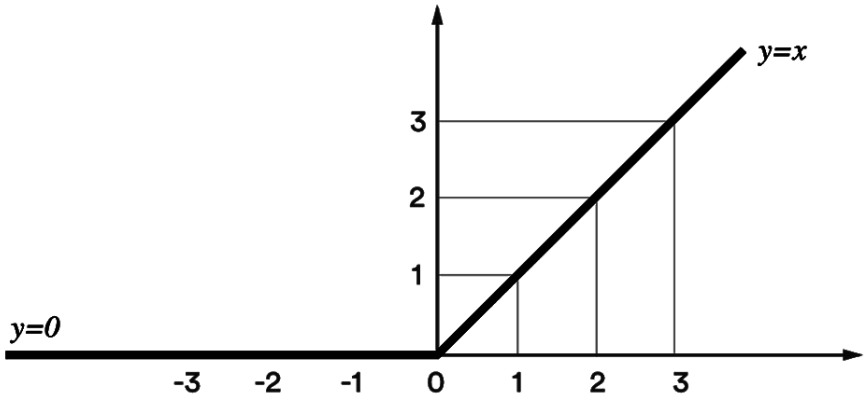


Рисунок 3.3 – Функція активації RELU

Для побудови архітектури згорткової нейронної мережі використовується три основні типи шарів: згортковий шар (англ. convolution layer), шар об'єднання або агрегації (англ. pooling layer) і повнозв'язний шар (точно так само, як і в звичайних нейронних мережах). Ці шари складаються для того, щоб сформувати повну архітектуру згорткової нейронної мережі.

Приклад архітектури згорткової нейронної мережі для класифікації цифр з датасету CIFAR-10 наведено на рисунку 3.4. Дана задача доволі подібна до задачі розпізнавання гестур, оскільки цифри можна розглядати як гестури.

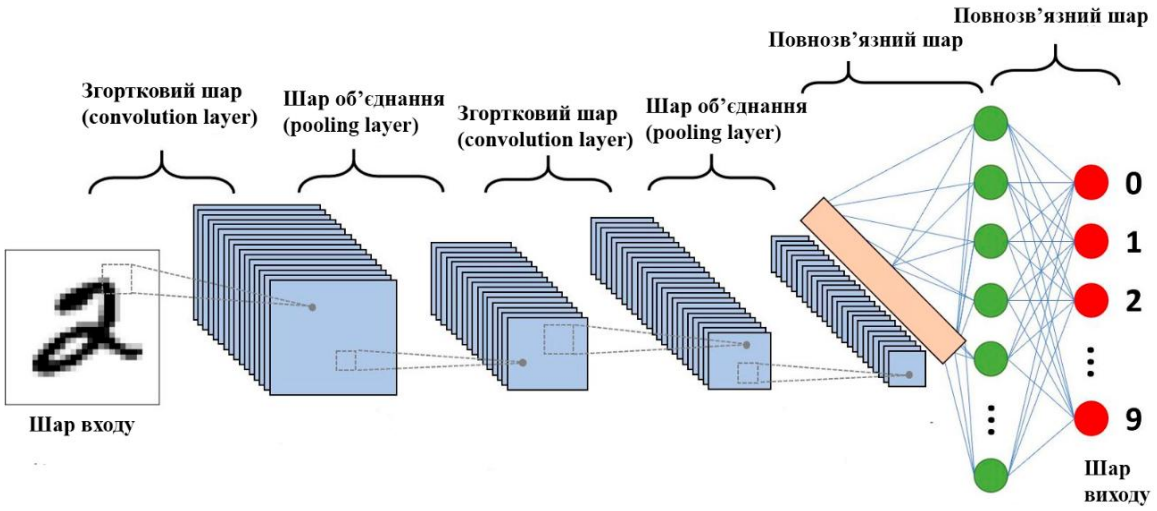


Рисунок 3.4 – Архітектура згорткової нейронної мережі

Шар входу буде отримувати необроблені значення пікселів зображення і буде представляти собою матрицю розмірності «ширина зображення» на «висота зображення» на «глибина зображення (кількість кольорових каналів)».

Згортковий шар є основним будівельним блоком згорткової нейронної мережі, який виконує більшу частину обчислень. Параметри згорткового шару складаються з набору фільтрів, які можна вивчити. Кожен фільтр невеликий просторово (по ширині і висоті), але проходить через повну

глибину вхідного обсягу. Наприклад, типовий фільтр на першому шарі згорткової нейронної мережі може мати розмір 5x5x3 (тобто ширину і висоту 5 пікселів і глибину 3, оскільки зображення мають 3 канали кольору). Під час руху вперед ми ковзаємо (точніше, згортаємо) кожен фільтр по ширині і висоті вхідного обсягу і обчислюємо скалярний добуток між записами фільтра і входу в будь-якому положенні. Коли ми пересуваємо фільтр по ширині та висоті вхідного обсягу, будемо створювати 2-мірну карту активації, яка дає відповіді на цей фільтр у кожному просторовому положенні. Інтуїтивно, мережа вивчатиме фільтри, які активуються, коли вони бачать певний тип візуальної функції, наприклад, край певної орієнтації або пляму деякого кольору на першому шарі, або в кінцевому підсумку цілісні стільникові або колесоподібні візерунки на вищих шарах мережі [12]. Тепер у нас буде весь набір фільтрів у кожному згортковому шарі (наприклад, 12 фільтрів), і кожен з них створить окрему 2-мірну карту активації. Ми будемо складати ці карти активації вздовж розмірності глибини і виробляти вихідний обсяг. Приклад обчислень згорткового шару можна побачити на рисунку 3.5.

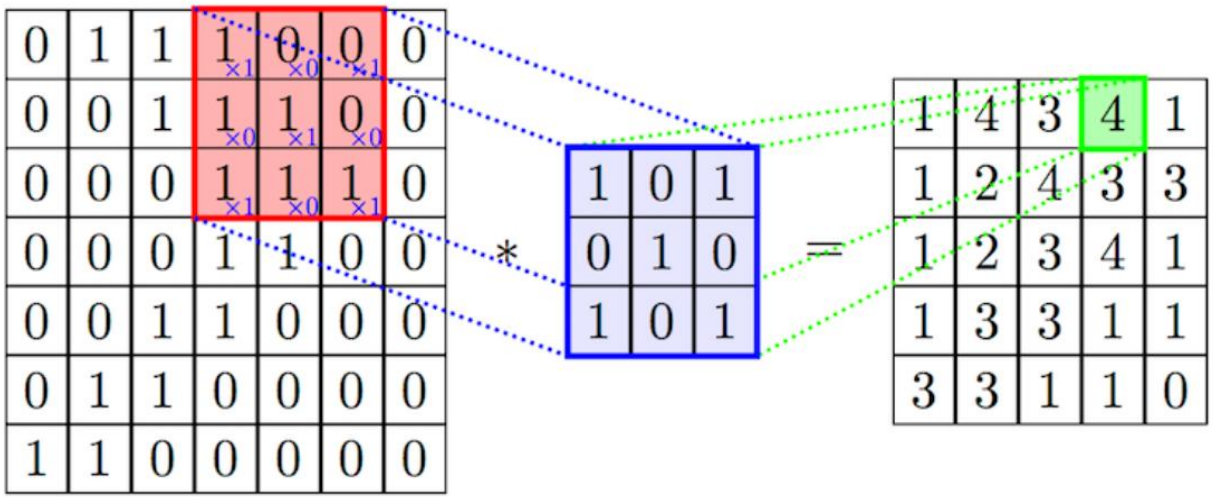


Рисунок 3.5 – Приклад обчислень згорткового шару

Шар об'єднання (агрегування) виконує операцію зменшення розміру по просторовим розмірам (ширина, висота). Він є одним із видів нелінійного зниження дискретизації. Використовуються декілька функцій для цього шару, найпоширенішою є максимізаційне агрегування (англ. max pooling), предствлене на рисунку 3.6.

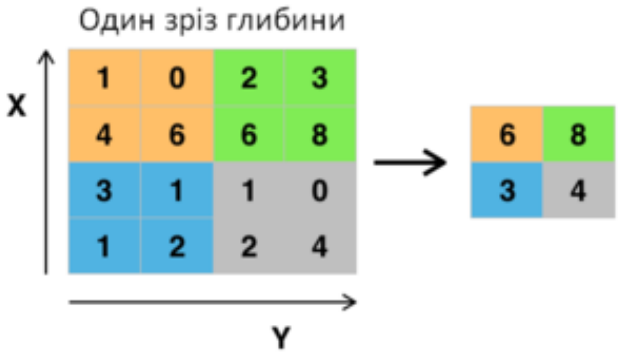


Рисунок 3.6 – Максимізаційне агрегування (англ. max pooling)

Повнозв'язний шар обчислює оцінки (бали) класів, в результаті чого на шару виходу у нас 10 нейронів, де кожен відповідає оцінці класу, наприклад, серед 10 цифр від 0 до 9 з датасету CIFAR-10. Як і звичайні нейронні мережі, і як випливає з назви, кожен нейрон в цьому шарі буде з'єднаний з усіма нейроні в попередньому шарі.

Таким чином, згорткові нейронні мережі шар за шаром перетворюють шар входу з зображенням на шар виходу зі значеннями кінцевих результатів кожного класу. Зауважте, що деякі шари містять параметри. Зокрема, шари згортки та повнозв'язні шари виконують перетворення функцією активації та векторний добуток параметрів (ваги нейронів). З іншого боку, шари згортки та об'єднання реалізують фіксовану функцію. Параметри в шарах згортки та повнозв'язних шарах будуть навчені градієнтним спуском, так що оцінки класів, які обчислюються згортковою нейронною мережею, узгоджуються з мітками в навчальному наборі для кожного зображення [13].

При роботі з великогабаритними входами, такими як зображення, як ми бачили вище, непрактично зв'язувати нейрони з усіма нейронами в попередньому обсязі. Замість цього ми з'єднаємо кожний нейрон лише з локальною областю вхідного обсягу. Просторова протяжність цього зв'язку є гіперпараметром, що називається рецептивним полем нейрона (еквівалентно розміру фільтра). Ступінь зв'язку по осі глибини завжди дорівнює глибині вхідного обсягу. Важливо ще раз підкреслити цю асиметрію в тому, як ми розглядаємо просторові розміри (ширину і висоту) і глибину: зв'язки локальні в просторі (уздовж ширини і висоти), але завжди повні по всій глибині вхідного об'єму [14].

3.4 Опис методів розв’язання

У даному дипломному проекті було використано алгоритм згорткової нейронної мережі. Саму модель було створено за допомогою бібліотеки Keras, яка дозволяє легким способом створити нейронну мережу будь якої архітектури та налаштовувати будь-які гіперпараметри. Також важливо зауважити що Keras оснований на TensorFlow – бібліотеці для глубокого навчання створеної у Google.

Оскільки на вході у нас зображення гестур, то використовувати ми будему згорткову нейронну мережу. Вхідне зображення, створене у відповідному вікні розміром 500x500 пікселі спочатку буде зменшене до 100x100 пікселів. Потім програма створить ще декілька схожих зображень методом аугментації зображень (невеликі зсуви початкового зображення вгору, вниз, вправо та вліво). Далі, проходячи через згортковий шар (з функцією активації RELU) та шар агрегації, вхідне зображення буде зменшувати звою розмірність, не втрачаючи своїх особливостей (певних горизонтальних та вертикальних ліній тощо), на основі яких нейронна мережа виконує розпізнавання. Таким чином, ми значно зменшили кількість нейронів і наша мережа може виконувати поставлену задачу за набагато

менший час і з більшою точністю, порівнюючи зі звичайними нейронними мережами. В кінці мережі ми маємо повнозв'язний шар, який переходить у шар виході, в якому все ми зможемо отримувати оцінки відповідності введеної користувачем гестури на розпізнавання з наявними в системі гестурами.

Висновок до розділу

У розділі математичного забезпечення була сформульована змістовна та математична постановки задачі, описано причини вибору даного способу розв'язання та його переваги. Було проаналізовано різницю між вирішенням поставленої задачі звичайною нейронної мережею та згортковою нейронною мережею, та описано її властивості, особливості та чому було вибрано другий варіант. Також був детально розібраний процес формування архітектури ЗНМ, описано всі наявні шари (згортковий, агрегувальний, повнозв'язний) та їх функції, процеси, які в цих шарах відбуваються, описано процес навчання згорткової нейронної мережі для розпізнавання гестур. Було визначено функції активації, наявні в системі (softmax для шару виходу та RELU для всіх інших шарів, де присутня функція активації), описано методи аугментації датасету (зображень гестур), за допомогою яких є можливість покращити точність роботи згорткової нейронної мережі.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Для побудови діаграм було використано такий засіб як Enterprise Architect. Для реалізації проекту було обрано мову програмування Python. За допомогою наведених нижче бібліотек, було створено графічний віконний інтерфейс, створено і натреновано ЗНМ.

Python - широко використовувана мова програмування високого рівня для програмування загального призначення. Крім мови програмування з відкритим вихідним кодом, python є чудовою об'єктно-орієнтованою, інтерпретованою та інтерактивною мовою програмування. Він поєднує в собі чудову потужність з дуже чітким синтаксисом. Python має модулі, класи, винятки, динамічні типи даних високого рівня і динамічну типізацію. Існують інтерфейси до багатьох системних викликів і бібліотек, а також до різних віконних систем. Нові вбудовані модулі можуть бути легко написані на мові C або C ++ (або на інших мовах, залежно від обраної реалізації). Python також можна використовувати як мову розширення для програм, написаних на інших мовах, які потребують простих у використанні сценаріїв або інтерфейсів автоматизації. Якщо ми подивимося на філософію мови Python, можна сказати, що ця мова була побудована для найбільш комфортного читання і найменшої складності написання коду.

На даний момент, Python – основний інструмент для розробки нейронних мереж, оскільки в ньому присутні всі найпопулярніші та найфункціональніші бібліотеки для аналізу даних, обробки та маніпуляції з датасетами, створення та налаштування всіх популярних та широко використовуваних алгоритмів машинного навчання та роботи з нейронними мережами різних архітектур (TensorFlow від Google, Keras, Theano, PyTorch, NumPy, Scikit-learn, SciPy).

Для розробки даного програмного забезпечення було використано Python версії 3.6, пакетний менеджер pip та такі бібліотеки:

- Keras – бібліотека, що містить всі необхідні класи та функції для побудови нейронних мереж (у тому числі ЗНМ) довільної архітектури. Також вона має багато корисних функцій, наприклад, для попередньої обробки датасета;
- Tensorflow – бібліотека для нейронних мереж від Google, на основі якої працює бібліотека Keras. На момент написання роботи, дана бібліотека не підтримує версію Python вище за 3.6, саме тому в даній дипломній роботі не використовується найновіша версія язика Python [15];
- Tkinter – бібліотека для створення графічного віконного інтерфейсу. За допомогою даної бібліотеки був створених весь графічний інтерфейс програмного продукті;
- pygame – бібліотека в основному використовується для створення ігор, але в даній роботі за допомогою pygame було реалізовано вікно, в якому користувач має вводити гестури (білий квадрат розміром 500x500 пікселів, на якому користувач малює чорним кольором гестури). З точки зору реалізації малювання, програма створює прямі лінії між положеннями миші користувача при зажатій лівій клавіши миші;
- PIL – бібліотека для роботи з зображеннями, використовується для роботи з гестурами, які зберігаються в виді файлів з розширенням «PNG» для збереження та відкриття таких файлів;
- re – бібліотека для роботи з регулярними виразами. В даному програмному продукті використовується для перевірки введеної користувачем назви гестуру на відповідність правилам введення назви (використовуються тільки великі та малі літери англійського алфавіту, цифри, дефіс та знак підкреслення);

- NumPy – одна з найпопулярніших бібліотек для математичних обчислень та роботи з масивами даних. Вона має велику кількість функцій, призначених для роботи з масивами спеціального типу «numpy array». Основні функції написані на мові C, то бібліотека досить швидко працює в порівнянні зі звичайними масивами Python. Ця бібліотека необхідна в даній роботі тому, що зображення гестур, які подаються на згорткову нейронну мережу, представлені у вигляді матриці, де значення елементів матриці є значеннями відповідних пікселів;
- os – для доступу до каталогу «gestures» (де зберігаються зображення гестур) щоб на основі зображень тренувати згорткову нейронну мережу та створювати вікно програми зі списком всіх наявних гестур та можливості ці гестури видаляти по бажанню користувача. Бібліотека працює швидко, що дає можливість, враховуючи специфіку вхідних та вихідних даних даного програмного забезпечення, не використовувати БД при розробці системи.

Прототипування та розробка згорткової нейронної мережі для даного програмного забезпечення було виконано в Jupyter Notebook з пакету Anaconda. Розробка інтерфейсу програми та всього іншого функціоналу, а також фінальна версія реалізації була виконана в інтегрованому середовищі розробки PyCharm від JetBrains. В цьому ж середовищі відбувалось тестування програмного продукту. Вся розборка, тестування та використання програмного продукту відбувалися в операційній системі Windows 10. БД не використовувалася по наведеним вище причинам, а доступ до файлів зображень гестур відбувався через стандартну бібліотеку Python – os, яка вміє працювати з каталогами в ОС Windows 10.

4.2 Вимоги до технічного забезпечення

Для коректного функціонування системи було виділено наступні вимоги до технічного забезпечення.

До складу технічних засобів повинні входити:

- комп’ютер з такою конфігурацією:
  1) процесор з тактовою частотою не нижче 1.5 ГГц;
  2) 32/64 розрядна операційна система;
  3) достатній об’єм оперативної пам’яті (не менше 2 Гб);
  4) інші складові можуть мати будь-які параметри, тому що вони не значним чином впливають на роботу програми;
- додатково має бути встановлене таке програмне забезпечення:
  1) Python v3.6;
  2) бібліотеки Keras, Tensorflow, Tkinter, pygame, PIL, NumPy;
- комп’ютерна периферія, до складу якої входить:
  1) монітор;
  2) мишка;
  3) клавіатура.

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма послідовності

Програмний продукт можна поділити на три основні процеси.

Перший процес – створення гестури, в якому користувачу потрібно ввести назву нової гестури та намалювати декілька прикладів зображень даної гестури.

Другий процес – редагування гестур. Користувач отримує список гестур з мініатюрами зображень та має змогу або видалити гестуру по його вибору, або додати до первої гестури приклад зображення, що призведе по

збільшення навчаючої виборки, а отже і до покращення роботи згорткової нейронної мережі.

Третій процес – саме розпізнавання гестур. Користувач вводить (малює) гестуру і після цього система проганяє отримане зображення і видає результат: гестуру буде розпізнано і у відповідному місці з'явиться текст з назвою цієї гестури, або система виведе повідомлення про те, що гестуру не розпізнано.

Для всіх перелічених вище процесів у графічному матеріалі наведено схеми структурні послідовності.

4.3.2 Діаграма розгортання

У графічному матеріалі наведено схему структурну розгортання. Як видно зі схеми, програма працює локально та незалежно в операційній системі користувача. Для роботи програми необхідно мати встановлений пакет Python 3.6 та всі необхідні бібліотеки, які були перелічені в пункті 4.2.

4.3.3 Специфікація функцій

Специфікацію функцій, які використовуються у даному програмному забезпеченні, наведено у таблиці 4.1.

Таблиця 4.1 – Функції програмного забезпечення

Назва	Опис
menu()	Дана функція створює головне меню програми. Саме вікно інтерфейсу свторюється за межамі цієї функуції. Дана функція створює всі кнопки на тест на головному меню програми, тобто кнопки «Recognize gesture», «Create new gesture», «List of gestures», «Exit». Також функція виводить текст з інформацією про назву програми, автора, та у спеціальному місці виводить назву розпізнаної гестури або тест про те, що введену користувачем гестуру не розпізнано

Продовження таблиці 4.1

recognize_gesture()	Дана функція спочатку викликає функцію draw_gesture(gesture_name), в яку передає текст «gesture_to_recognize». Ця функція, в свою чергу, створює окреме вікно, де користувач може напалювати гестуру. Після введення користувачем гестури, функція recognize_gesture() виконує деякі зміни в тільки но створеному зображенні: завантажує зображення в оперативну пам’ять, робить зображення чорні-білим, перетворює зображення на спеціальний масив (об’єкт бібліотеки numpy), проводить нормалізацію масиву, змінює розмірність масиву на таку, яка необхідна для відправки на модель згорткової нейронної мережі (об’єкт бібліотеки keras). Потім масив відправляється на опрацювання в нейронну мережу, яка повертає значення оцінок мережі на відповідність введеної гестури до існуючих в системі гестур. Програма вибирає найбільшу оцінку и відповідна назва гестури виводиться в головному меню програми. Якщо всі оцінки згорткової нейронної мережі менше за порогового значення, то система виводить повідомлення про те, що гестуру не розпізнано
create_gesture()	Дана функція видаляє всі існуючі кнопки на текстові поля на головному меню програми, та створює кнопки та тестові поля, необхідні для вікна створення нової гестури, тобто кнопки «Confirm» и «Menu», а також текст про те, які символи можна використовувати при створення назви нової гестури та текстове поле, куди користувач може вводити назву гестури, яку він хоче створити

Продовження таблиці 4.1

confirm()	Дана функція перевіряє введену користувачем назву нової гестури на відповідність до символів, які можна використовувати при створенні нових гестур. Якщо є невідповідні символи, то функція робить червоним тест, в якому написано, які символи можна використовувати. Якщо всі символи вірні, то функція заміняє кнопку «Confirm» на кнопку «Add gesture sample», натиснувши на яку користувач визове функцію draw_gesture(gesture_name), в яку передасть відповідну назву гестури
draw_gesture(gesture_name)	Функція отримує на вхід назву гестури, яка вводиться користувачем (gesture_name). Дана функція створює окреме вікно з білим фоном, де користувач за допомогою миші або сенсорного екрану може напалювати гестуру чорною лінією довільної форми. Після введення користувачем гестури, користувач має закрити вікно або натиснути клавішу «Enter» на клавіатурі, після цього функція дивиться на каталог «gestures» для того, щоб визначити який по номеру це буде приклад зображення гестури для того, щоб правильно створити файл з зображенням, дописавши до назви гестури порядковий номер екземпляра (після символа підкреслення). Якщо назва гестури – «gesture_to_recognize», то файл з зображенням створюється у головному каталозі програми з назвою «gesture.png». Після створення зображення, функція змінює його розмірність до 100x100 пікселів. Це необхідно для більш точної роботи згорткової нейронної мережі. Початкова розмірність зображень – 500x500 пікселів

Продовження таблиці 4.1

scale_image(input_image_path, output_image_path,width,height)	Дана функція змінює розмір зображення. Вона отримує на вхід шлях до початкового зображення, яке потрібно змінити, кінечний шлях зміненого зображення (якщо він співпадає зі шляхом початкового зображення, то воно буде перезаписане), необхідну ширину і висоту кінечного зображення
list_of_gestures()	Дана функція видаляє всі існуючі кнопки на текстові поля на головному меню програми. Потім функція дивить у каталог «gestures» та бере звідти список всіх існуючих гестур у системі
show_list(n)	Функція отримує на вхід порядковий номер гестури, з якої буде починатися список (n). Це необхідно тому, що на вікно зі списком гестур поміщається тільки 3 гестури. Дана функція наповнює віно зі список гестур, тобто створює мініатюрне зображення першого екземпляра гестури, її назву, кнопки «Add sample» (викликає функцію draw_gesture(gesture_name), в яку передає назву відповідної гестури), «Delete gesture» (викликає функцію delete_gesture(name), в яку передає назву відповідної гестури) для кожної з 3 гестур, розміщених у даному вікні. Також, для ситуацій коли гестур більше 3, існують кнопки «Prev» та «Next», за допомогою яких можна перелистувати сторінки. Ще є кнопка «Menu», яка переводить користувача до головного меню програми
delete_gesture(name)	Функція отримує на вхід назву гестури, яку потрібно видалити. Дана функція видаляє всі зображення з каталогу «gestures», які відносяться до видаляємої гестури та визиває функцію train() для повторного навчання згорткової нейронної мережі на новому наборі даних

Продовження таблиці 4.1

close()	Дана функція визиває функцію save_weights() для того, щоб зберегти поточні значення ваг згорткової нейронної мережі, а потім закриває програму
train()	Дана функція запускає процес тренування згорткової нейронної мережі на зображеннях гестур, присутніх в каталозі «gestures». Також вона змінює кількість нейронів на шарі виходу – їх кількість повинна бути також же, як і кількість наявних в системі гестур

Висновок до розділу

У розділі програмного та технічного забезпечення було описано засоби розробки, що використовувалися для створення програмного продукту, вимоги до технічного забезпечення, виконання яких необхідне для того, щоб запустити систему на своєму комп’ютері. Також було описано архітектуру програмного забезпечення, що включає в себе діаграми послідовності, які описують дії користувача в системі, діаграму розгортання та специфікацію функцій даного програмного продукту, в якій детально розписані всі наявні в системі функції, що вони отримують на вхід, що роблять та який результат видають.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Користувач для початку роботи з програмним забезпеченням повинен запустити програму. Після завантаження програми, користувач потрапляє на головне вікно програми – меню, яке можна побачити на рисунку 5.1.

Рисунок 5.1 – Головне меню програми

Розглянемо детальніше головне меню програми:

1) запуск окремого вікна, на якому можна ввести гестуру, яка буде розпізнаватися системою. Результат розпізнавання система виведе у відповідному місці;
2) перехід у вікно створення нової гестури;
3) перехід у вікно списку існуючих гестур;
4) вихід з програми;
5) місце виведення результатів розпізнавання програмою введеної користувачем гестури;
6) назва програми та автор.

Перейдемо у вікно створення нової гестури, яке можна побачити на рисунку 5.2.

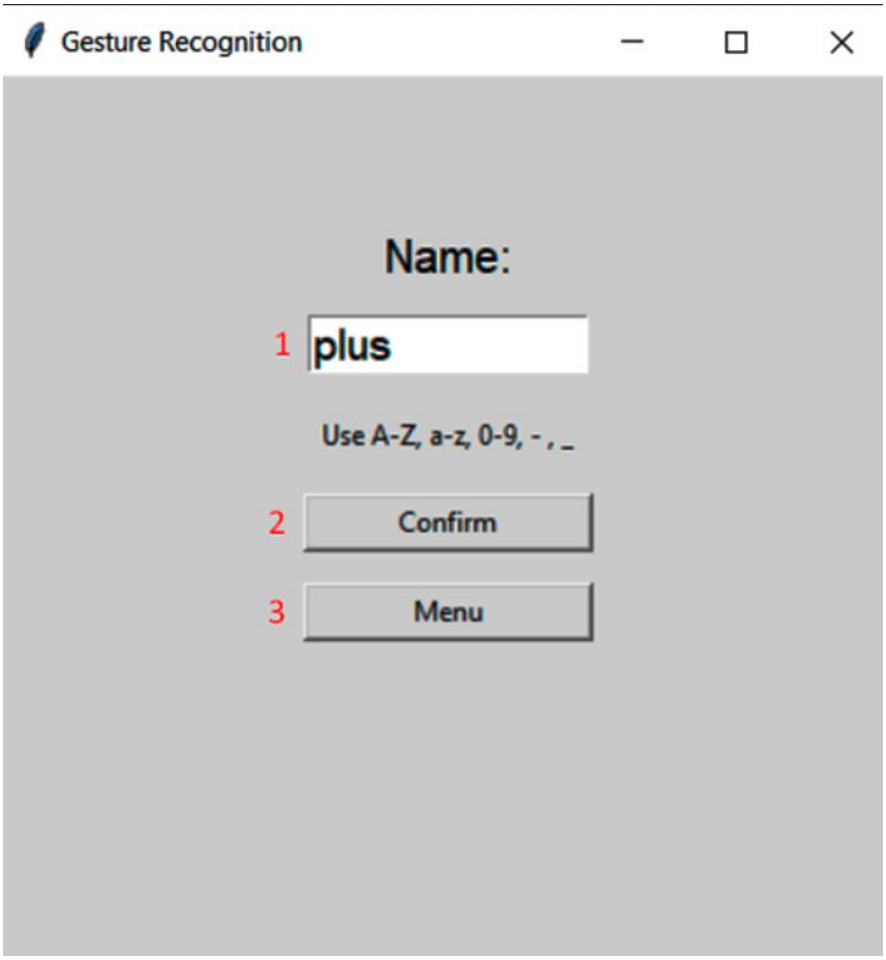


Рисунок 5.2 – Вікно створення нової гестури

Розглянемо наявні елементи в даному вікні програми:

1) назва нової гестури. Користувач сам придумає назву. Можна використовувати тільки великі та малі літери англійського алфавіту, цифри, дефіс та знак підкреслення. У випадку використання інших символів, система не дозволить створити нову гестуру;

2) кнопка, яке зберігає введену користувачем назву нової гестури та дозволяє створювати приклади даної гестури;

3) повернення назад до меню.

Після підтвердження назви гестури, система дає змогу користувачу додати до даної гестури будь яку кількість прикладів зображень цієї гестури. Зробити це можна натиснувши на відповідну кнопку, яку можна побачити на рисунку 5.3. Для створення гестури, потрібно додати хоча б одне зображення гестури.

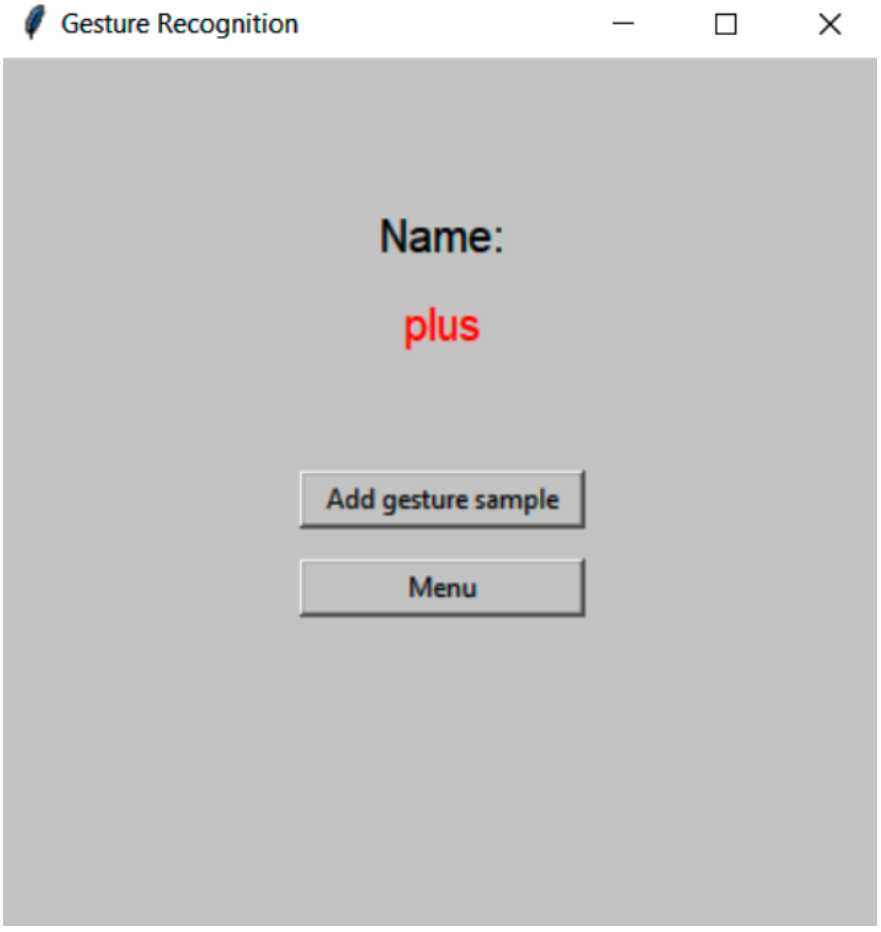


Рисунок 5.3 – Вікно створення нової гестури

Після натискання на кнопку «Add gesture sample», відкриється нове вікно, на якому користувач може намалювати гестуру, який буде збережено до каталогу «gestures» з поміткою, що ця гестура має введену користувачем назву (в даному випадку це «plus»). Користивач може добавляти зображення прикладу гестур скільки завгодно, це тільки збільшить точність роботи згортковою нейронної мережі (це особливо важливо при наявності в системі дуже схожих гестур, для точного розпізнавання яких згортковій нейронній мережі необхідно більше прикладів зображень гестур). Бажано, щоб було принаймні 3 зображення кожної гестури, але ця кількість необмежена і користувач додає приклади гестур у вибронй ним кількості. Вікно створення зображення гестури можна побачити на рисунку 5.4.

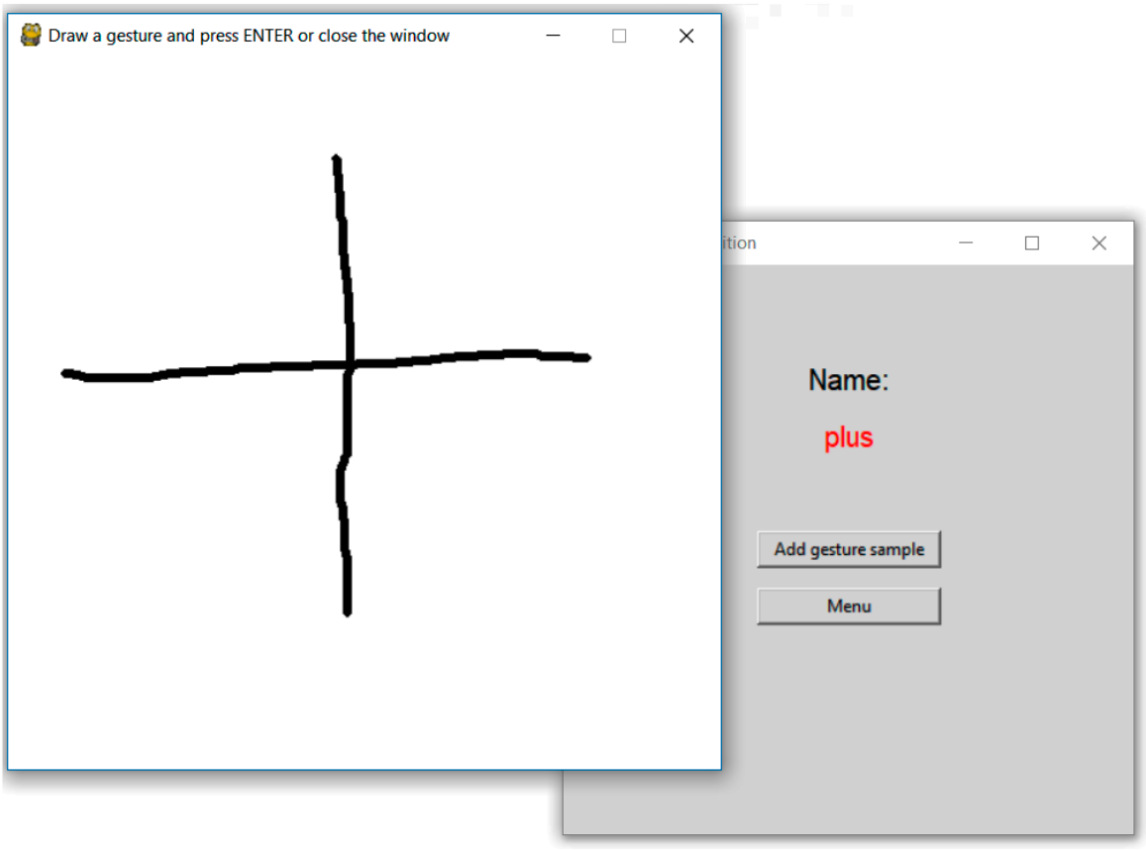


Рисунок 5.4 – Вікно створення зображення гестури

Після того, як користувач закінчив вводити гестуру, він повинен натиснути на клавіатурі клавішу «Enter» або просто закрити вікно (мишкою або комбінацією клавіш «Alt»+«F4»). Система сама збереже введену гестуру в каталозі «gestures» та дасть їй правильну назву файла.

Перейдемо до вікна зі списком усіх наявних гестур в системі. На рисунку 5.5 видно список гестур, їх малюнки, назву та кнопки керування.

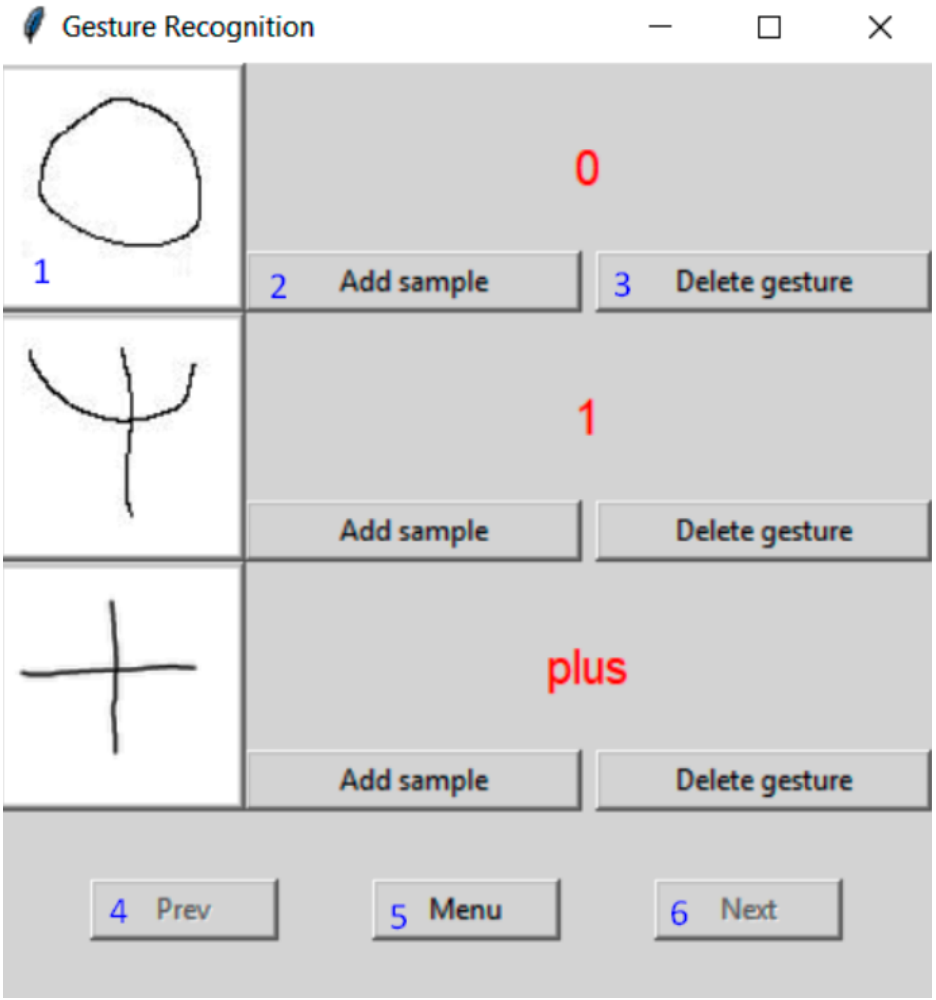


Рисунок 5.5 – Вікно списку гестур

Як видно з рисунка, в системі наявно 3 різних гестури. Розглянемо детальніше вікно списку гестур:

1) зображення гестур (береться самий перший екземпляр);

2) додати новий приклад зображення гестури до вибраної існуючої гестури;

3) видалити гестуру (тим самим видаляться всі зображення гестури з каталогу «gestures»);

4) перегорнути сторінку назад;

5) повернутися назад до меню;

6) перегорнути сторінку вперед.

Розглянемо як працює розпізнавання гестур. При натисканні на кнопку «Recognize gesture» з’являється вікно створення зображення гестури. Після введення гестури, користувач повинен натиснути на клавіатурі клавішу «Enter» або просто закрити вікно (мишкою або комбінацією клавіш «Alt»+«F4»). Далі у головному меню програми у відповідному місці з’явиться назва гестури, якщо система його розпізнала, або повідомлення про те, що гестуру не розпізнано. На рисунку 5.6 було введено гестуру «plus» і система його розпізнала. Потім було ще раз введеного гестуру «plus» для наглядності рисунка.

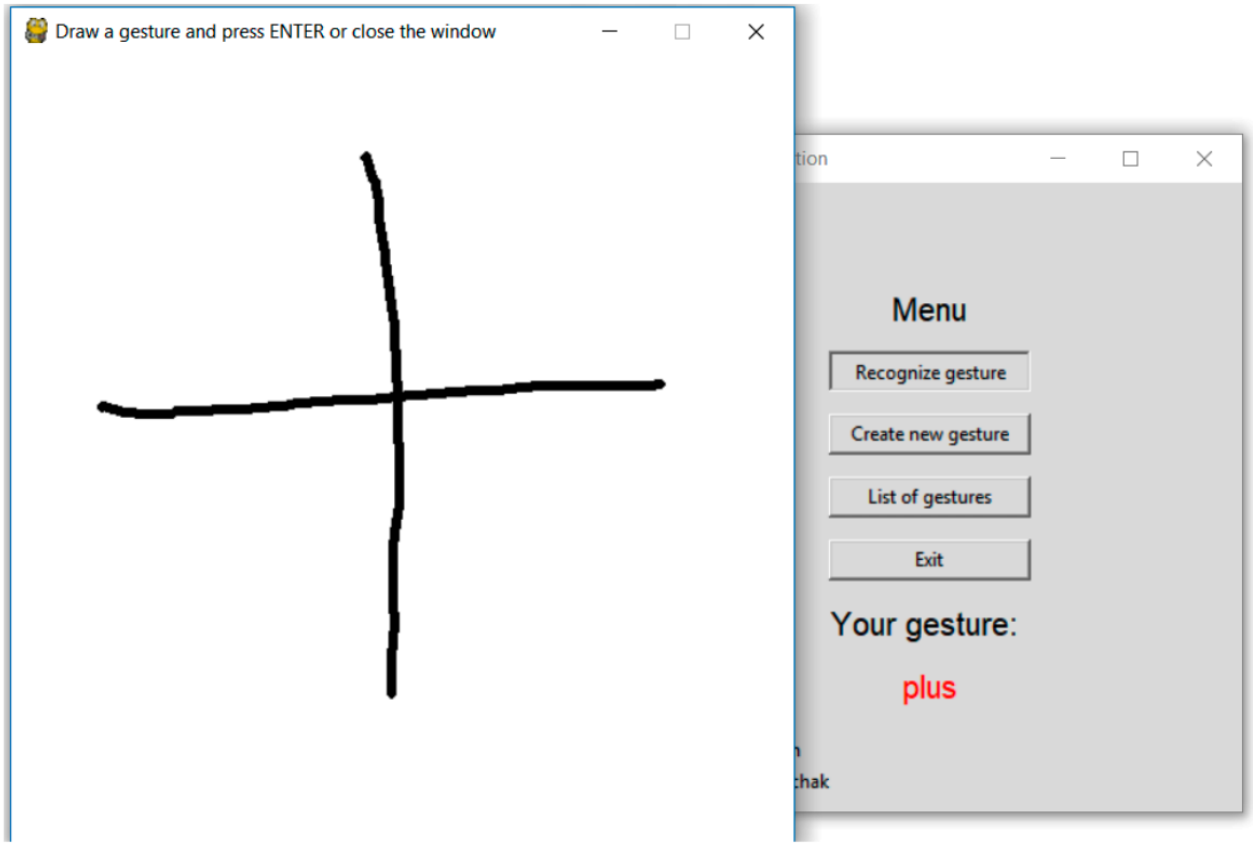


Рисунок 5.6 – Розпізнавання гестури

5.2 Випробування програмного продукту

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій інформаційної системи підтримки розпізнавання комп'ютерних гестур вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

- ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
- ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

Розглянемо основні функції системи та перевіримо їх відповідність вимогам технічного завдання. У таблицях 5.1 – 5.9 наведено перелік випробувань основних функціональних можливостей програмного засобу.

Таблиця 5.1 – Перевірка можливості відкриття вікна введеня гестури для розпізнавання

Початковий стан	Відкрите головне меню програми
Схема проведення тесту	Натиснути кнопку «Recognize gesture»
Очікуваний результат	Відкриється вікно введеня гестури, яке представляє собою палітру розміром 500x500 пікселів, де користувач може мишкою або рукою (у разі сенсорного екрану) намалювати гестуру чорною лінією
Стан після випробування	Відкрито вікно введеня гестури

Таблиця 5.2 – Перевірка можливості відкриття вікна створення нової гестури

Початковий стан	Відкрите головне меню програми
Схема проведення тесту	Натиснути кнопку «Create new gesture»
Очікуваний результат	Відкриється вікно створення нової гестури, в якому користувач повинен придумати та ввести назву нової гестури
Стан після випробування	Відкрито вікно створення нової гестури

Таблиця 5.3 – Перевірка введення назви нової гестури

Початковий стан	Відкрите сторінка створення нової гестури
Вхідні дані	Назва гестури, яку користувач бажає створити в системі
Схема проведення тесту	Ввести обрану користувачем назву гестури у відповідне поле в вікні програми та натиснути кнопку «Confirm» для підтвердження вводу.
Очікуваний результат	Система прийме введену назву та дасть можливість користувачу вводити самі гестури (малювати зображення гестур) натискаючи на кнопку «Add gesture sample».
Стан після випробування	Система прийняла назву гестури, та дала можливість користувачи додавати приклади гестур, натискаючи на кнопку «Add gesture sample».
У разі помилки	Підсвічується червоним текстове поле, в якому вказано, що для створення назви гестури потрібно використовувати великі та малі літери англійського алфавіту, цифри, дефіс або підкреслення.

Таблиця 5.4 – Перевірка додавання нових прикладів гесур до створеної гестури

Початковий стан	Відкрита сторінка створення нової гестури з введеною та підтвердженою назвою гестури
Вхідні дані	Зображення гестури
Схема проведення тесту	Користувач натискає на кнопку «Add gesture sample» та малює гестуру у відповідному вікні
Очікуваний результат	Система зчитує зображення та зберігає його в каталозі «gestures», давши йому відповідну до назви гестури назву файлу
Стан після випробування	Система зчитала зображення та зберегла його в каталозі «gestures», давши йому відповідну до назви гестури назву файлу

Таблиця 5.5 – Перевірка можливості відкриття вікна зі списком наявних в системі гестур

Початковий стан	Відкрите головне меню програми
Схема проведення тесту	Натиснути кнопку «List of gestures»
Очікуваний результат	Відкриється вікно зі списком наявних в системі гестур, в якому користувач може переглянути всі створені ним гестури, додати нові приклади введення гестур або видалити певні гестури. Вікно вміщає тільки 3 гестури, тому наявні кнопки навігації по спику: «Prev» та «Next»
Стан після випробування	Відкрито вікно зі списком наявних в системі гестур, на якому користувач може додавати нові приклади гестур або видаляти гестури з системи. Оскільки вікно вміщає тільки 3 гестури, то наявні кнопки навігації по спику: «Prev» та «Next»

Таблиця 5.6 – Перевірка можливості додавання нових прикладів гестур зі списку гестур

Початковий стан	Відкрите вікно програми зі списком усіх наявних в системі гестур
Схема проведення тесту	Натиснути кнопку «Add sample» біля вибраної гестури, у новому вікні ввести гестуру та закрити вікно або натиснути клавішу «Enter» на клавіатурі
Очікуваний результат	Після натискання на кнопку «Add sample» біля вибраної гестури, відкривається нове вікно введення гестур. Після того, як користувач введе гестуру та закриє вікно або натисне клавішу «Enter» на клавіатурі, система зчитує зображення та зберігає його в каталозі «gestures», давши йому відповідну до назви гестури назву файлу
Стан після випробування	Відкрилось нове вікно введення гестур. Після того, як користувач ввів гестуру та закрив вікно, система зчитала зображення та зберегла його в каталозі «gestures», давши йому відповідну до назви гестури назву файлу

Таблиця 5.7 – Перевірка можливості видалення гестур

Початковий стан	Відкрите вікно програми зі списком усіх наявних в системі гестур
Схема проведення тесту	Натиснути кнопку «Delete gesture» біля вибраної гестури
Очікуваний результат	Після натискання на кнопку «Delete gesture» біля вибраної користувачем гестури, система видаляє всі приклади гестури в каталозі «gestures», що призводить до видалення гестури з системи
Стан після випробування	Система видалила всі приклади вибраної на видалення гестури в каталозі «gestures», що призвело до видалення даної гестури з системи

Таблиця 5.8 – Перевірка правильності розпізнавання гестур

Початковий стан	Система в якій наявна гестура «plus», наведена на рисунку 5.7
Схема проведення тесту	Натиснути кнопку «Recognize gesture» в головному меню та ввести (намалювати) гестуру, подібну до гестури «plus»
Очікуваний результат	Система виведе текст, у якому буде вказано назву розпізнаної гестури («plus»)
Стан після випробування	Система розпізнала введену гестуру, подібну по гестури «plus», яка уже наявна в системі, і вивела повідомлення з назвою розпізнаної гестури, яке можна почати на рисунку 5.8

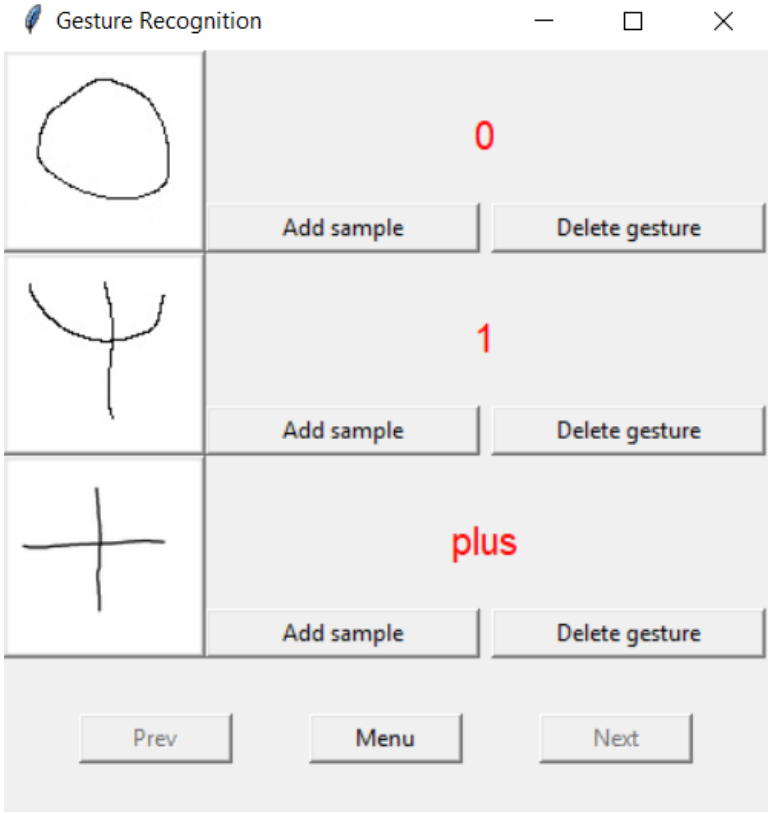


Рисунок 5.7 – Гестура «plus» у списку гестур

Рисунок 5.8 – Гестура, яку розпізнала система

Таблиця 5.9 – Перевірка правильності розпізнавання гестур

Початковий стан	Система в якій наявна гестура «plus», наведена на рисунку 5.7
Схема проведення тесту	Натиснути кнопку «Recognize gesture» в головному меню та ввести (намалювати) гестуру, яка відрізняється від гестури «plus»
Очікуваний результат	Система виведе текст, у якому буде повідомлення про те, що гестуру не розпізнано, оскільки такої гестури не має в системі
Стан після випробування	Система не розпізнала введену гестуру та вивела відповідне повідомлення, яке можна почати на рисунку 5.9

Рисунок 5.9 – Гестура, яку не розпізнала система

Висновок до розділу

У технологічному розділі було описано керівництво користувача, в якому зазначено які конкретно дії може виконувати користувач в системі і яким чином він повинен їх робити.

У підрозділі з випробуваннями програмного продукту було описано мету випробувань, загальні положення та наведено результати випробувань.

Метою випробувань являється перевірка відповідності функцій інформаційної системи підтримки розпізнавання комп'ютерних гестур вимогам технічного завдання.

Випробування проводилися шляхом функціонального тестування. Було описано основні функції системи та перевірено їх відповідність вимогам технічного завдання.

Для функціональних вимог було прописані окремі випробування, у яких присутні такі поля: початковий стан, вхідні дані, схема проведення тесту, очікуваний результат, стан після випробування, результат у разі помилки. Також було проведено випробування основного функціоналу програмного продукту – розпізнавання гестур. Було розпізнано гестуру, яка вже була наявна в системі. Гестуру, якої в системі не було, система не розпізнала та вивела відповідне повідомлення.

У результаті проведення випробувань було виправлено всі виявлені помилки.

ЗАГАЛЬНІ ВИСНОВКИ

У результаті виконання дипломного проекту було розроблено інформаційну систему підтримки розпізнавання комп'ютерних гестур.

Запропонований програмний продукт присвячений роботі з комп’ютерними гестурами, тобто з жестами, які користувач вводить на екрані комп’ютера за допомогою миші або пальця (у випадку сенсорного екрану).

У розділі опису програмного середовища було розглянуто предметне середовище, було визначено та описано процес діяльності, що наведений у структурній схемі діяльності. Іншим результатом є створення функціональної моделі системи: опис користувачів системи, їх ролі та можливі дії у системі. Функціональна модель представлена у вигляді структурної схеми варіантів використання.

У розділі інформаційного забезпечення було описано вхідні дані, які необхідні для правильної роботи системи. Оскільки алгоритм розпізнавання гестур – це згорткова нейронна мережа, то їй необхідна деяка кількість зображень гестур для того, щоб навчитися ці самі гестури розпізнавати надалі. Чим більший розмір датасету – тим краще і точніше алгоритм зможе розпізнавати гестури. Якщо деякі гестури будуть сильно схожі одна на одну, то для достатньої точності розпізнавання потрібно буде ввести дещо більше прикладів відповідних гестур. У разі частих помилок, у користувача завжди є можливість добавити нові приклади зображень гестур.

Вихідними даними системи є результати роботи згорткової нейронної мережі над введеною користувачем гестурою на розпізнавання. Система виводить або назву введеної гестури, якщо вона її розпізнала, або повідомлення про те, що гестуру не розпізнано.

Також у розділі було описано структуру зберігання даних. База даних не використовується за непотрібністю, оскільки система не зберігає ніяких масивів даних. Майже всі дані (крім вагів згорткової нейронної мережі у

файлі формату «h5») зберігаються у вигляді зображень з розширенням «PNG». Таким чином, у системи є всі необхідні методи для доступу до каталогу «gestures», де зберігаються зображення гестур, і використання бази даних призведе тільки до зменшення швидкості роботи системи, а також до надлишкових компонентів, що збільшить можливість виникнення проблем при використанні даного програмного продукту.

У розділі математичного забезпечення була сформульована змістовна та математична постановки задачі, описано причини вибору даного способу розв’язання та його переваги. Було проаналізовано різницю між вирішенням поставленої задачі звичайною нейронної мережею та згортковою нейронною мережею, та описано її властивості, особливості та чому було вибрано другий варіант. Також був детально розібраний процес формування архітектури ЗНМ, описано всі наявні шари (згортковий, агрегувальний, повнозв’язний) та їх функції, процеси, які в цих шарах відбуваються, описано процес навчання згорткової нейронної мережі для розпізнавання гестур. Було визначено функції активації, наявні в системі (softmax для шару виходу та RELU для всіх інших шарів, де присутня функція активації), описано методи аугментації датасету (зображень гестур), за допомогою яких є можливість покращити точність роботи згорткової нейронної мережі.

У розділі програмного та технічного забезпечення було описано засоби розробки, що використовувалися для створення програмного продукту, вимоги до технічного забезпечення, виконання яких необхідне для того, щоб запустити систему на своєму комп’ютері. Також було описано архітектуру програмного забезпечення, що включає в себе діаграми послідовності, які описують дії користувача в системі, діаграму розгортання та специфікацію функцій даного програмного продукту, в якій детально розписані всі наявні в системі функції, що вони отримують на вхід, що роблять та який результат видають.

У технологічному розділі було описано керівництво користувача, в якому зазначено які конкретно дії може виконувати користувач в системі і яким чином він повинен їх робити.

У підрозділі з випробуваннями програмного продукту було описано мету випробувань, загальні положення та наведено результати випробувань.

Метою випробувань являється перевірка відповідності функцій інформаційної системи підтримки розпізнавання комп'ютерних гестур вимогам технічного завдання.

Випробування проводилися шляхом функціонального тестування. Було описано основні функції системи та перевірено їх відповідність вимогам технічного завдання.

Для функціональних вимог було прописані окремі випробування, у яких присутні такі поля: початковий стан, вхідні дані, схема проведення тесту, очікуваний результат, стан після випробування, результат у разі помилки. Також було проведено випробування основного функціоналу програмного продукту – розпізнавання гестур. Було розпізнано гестуру, яка вже була наявна в системі. Гестуру, якої в системі не було, система не розпізнала та вивела відповідне повідомлення.

У результаті проведення випробувань було виправлено всі виявлені помилки.

Розроблений програмний продукт є корисним для розробників програмного забезпечення, оскільки дає їм готове рішення для використання функціоналу розпізнавання комп'ютерних гестур у своїх програмних продуктах.

У перспективі цей проект можна розвивати шляхом покращення точності розпізнавання, корегуючи гіперпараметри або змінюючи архітектуру згорткової нейронної мережі.

ПЕРЕЛІК ПОСИЛАНЬ

1. A Concise History of Neural Networks [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec
2. What is an artificial neural network? [Електронний ресурс]. – Режим доступу: https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/
3. Gesture recognition technology [Електронний ресурс]. – Режим доступу: http://www.jetir.org/papers/JETIR1701021.pdf
4. Architectural Innovations in Convolutional Neural Networks for Image Classification [Електронний ресурс]. – Режим доступу: https://machinelearningmastery.com/review-of-architectural-innovations-for-convolutional-neural-networks-for-image-classification/
5. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, 1989.
6. Convolutional Neural Networks (LeNet) [Електронний ресурс]. – Режим доступу: http://deeplearning.net/tutorial/lenet.html
7. Subject independent facial expression recognition with robust face detection using a convolutional neural network [Електронний ресурс]. – Режим доступу: http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/sparse/matsugo_etal_face_expression_conv_nnet.pdf
8. Convolutional neural network [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Convolutional_neural_network
9. How to Configure Image Data Augmentation When Training Deep Learning Neural Networks [Електронний ресурс]. – Режим доступу: https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/

10. Unicode characters table [Електронний ресурс]. – Режим доступу: https://www.rapidtables.com/code/text/unicode-characters.html
11. Convolutional Neural Network for Image Classification [Електронний ресурс]. – Режим доступу: https://pdfs.semanticscholar.org/b8e3/613d60d374b53ec5b54112dfb68d0b52d82c.pdf
12. About Convolutional Layer and Convolution Kernel [Електронний ресурс]. – Режим доступу: https://blog.sicara.com/about-convolutional-layer-convolution-kernel-9a7325d34f7d
13. Convolutional Neural Networks, Explained [Електронний ресурс]. – Режим доступу: https://www.datascience.com/blog/convolutional-neural-network
14. Convolutional Neural Networks (CNNs / ConvNets) [Електронний ресурс]. – Режим доступу: http://cs231n.github.io/convolutional-networks/
15. TensorFlow Documentation [Електронний ресурс]. – Режим доступу: https://buildmedia.readthedocs.org/media/pdf/tensorflow-object-detection-api-tutorial/latest/tensorflow-object-detection-api-tutorial.pdf

Додаток А

Тексти програмного коду

Інформаційна система підтримки розпізнавання комп'ютерних гестур

(Найменування програми (документа))

DVD-R

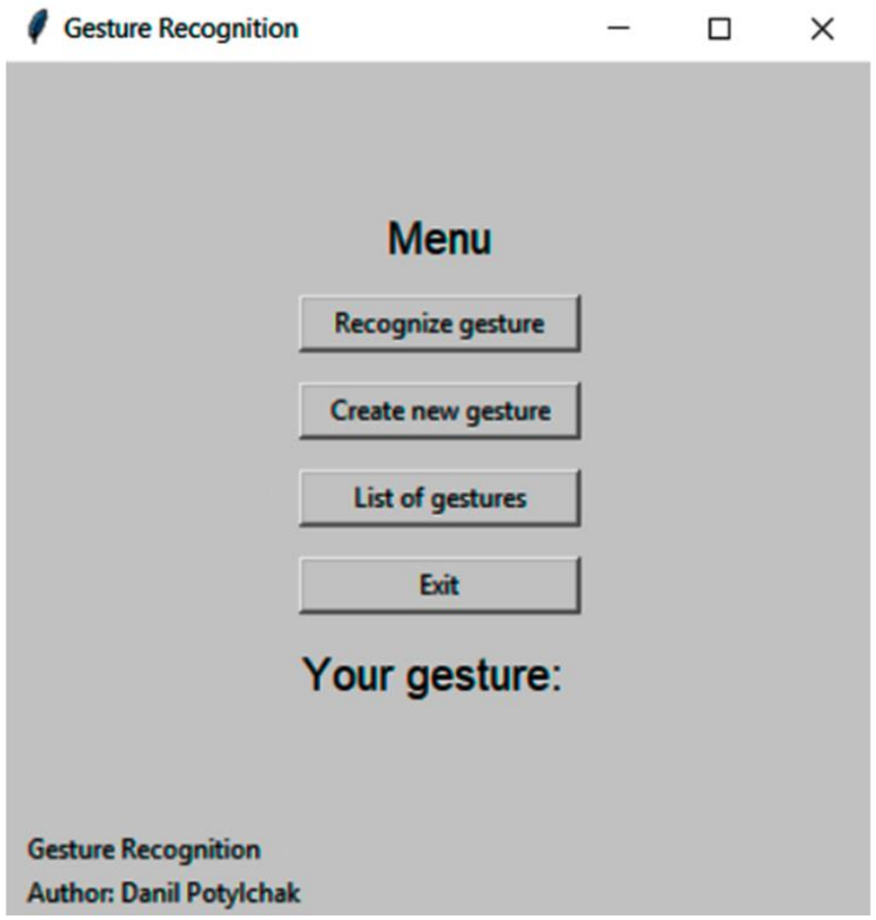
(Вид носія даних)

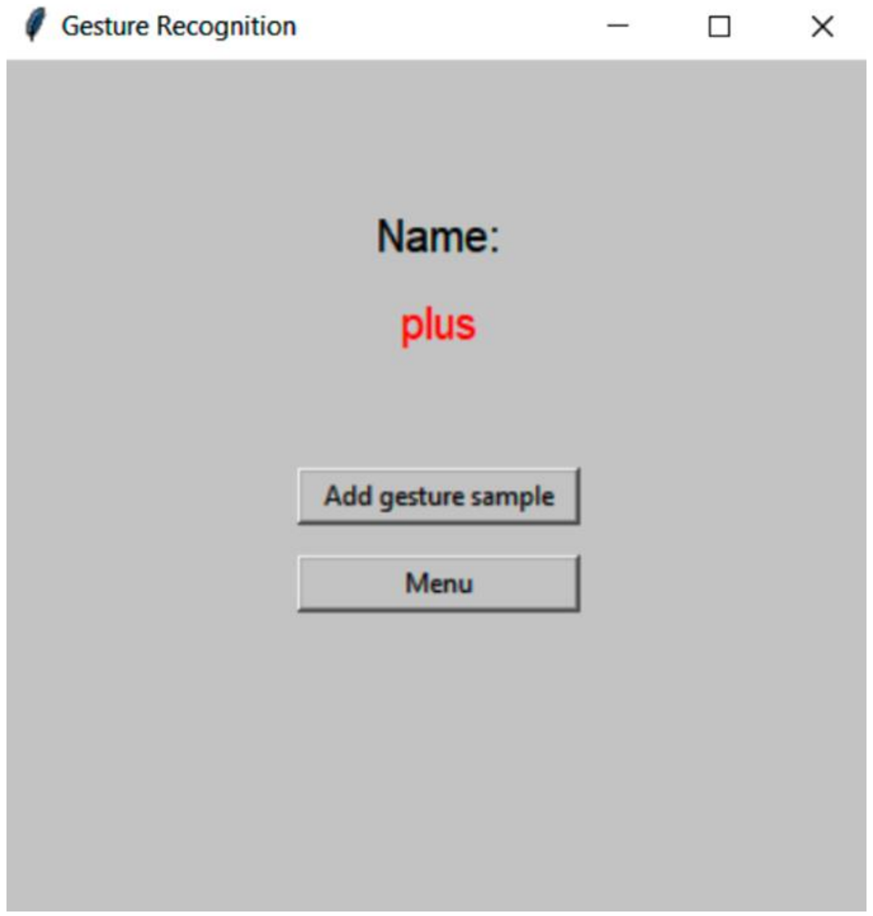
10 арк, 15,1 Кб

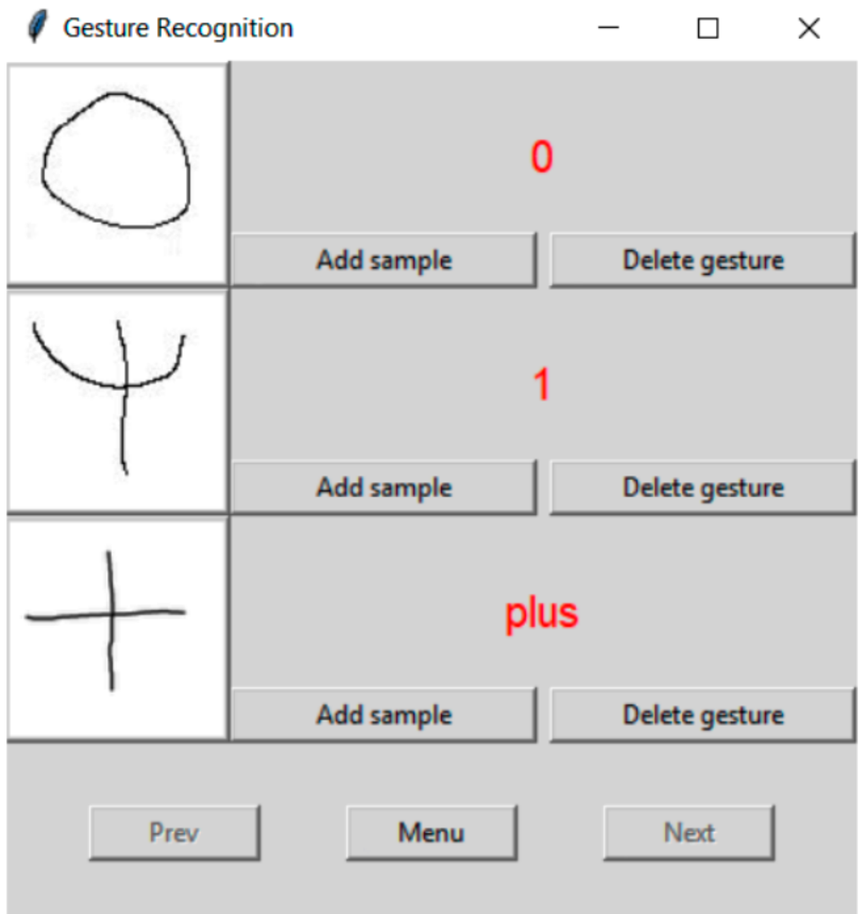
(Обсяг програми (документа) , арк.,) Кб)

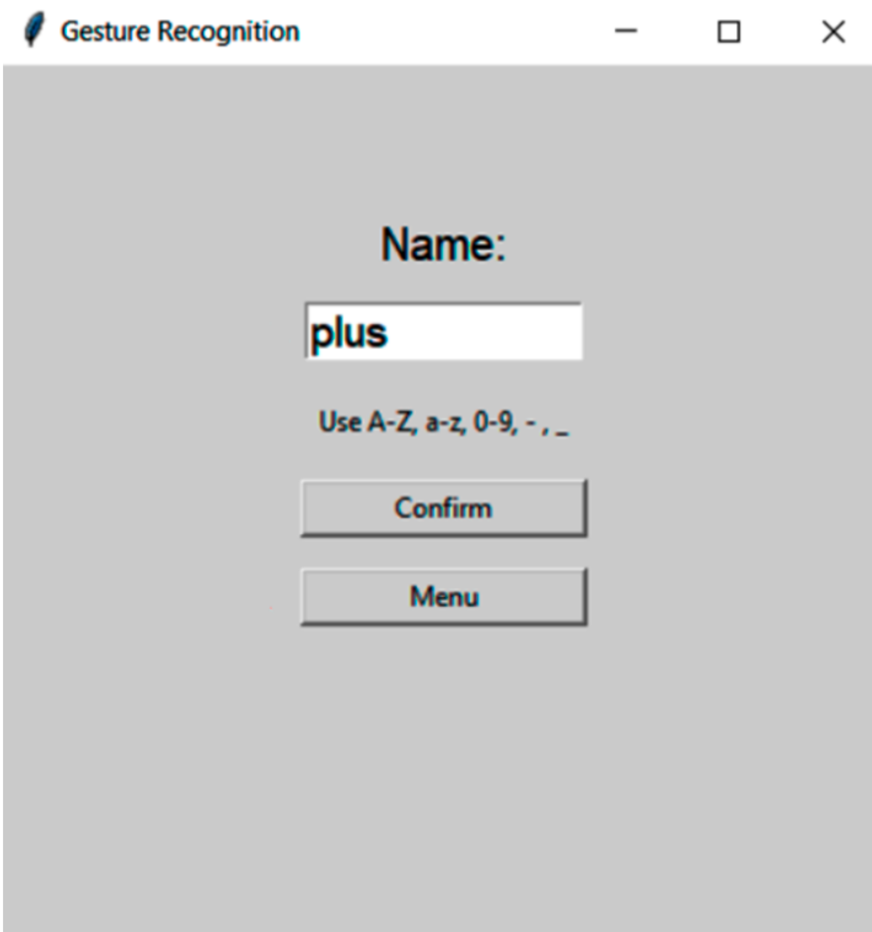
Київ – 2019 року

ДП ІС-5120.1181-с.КЕ

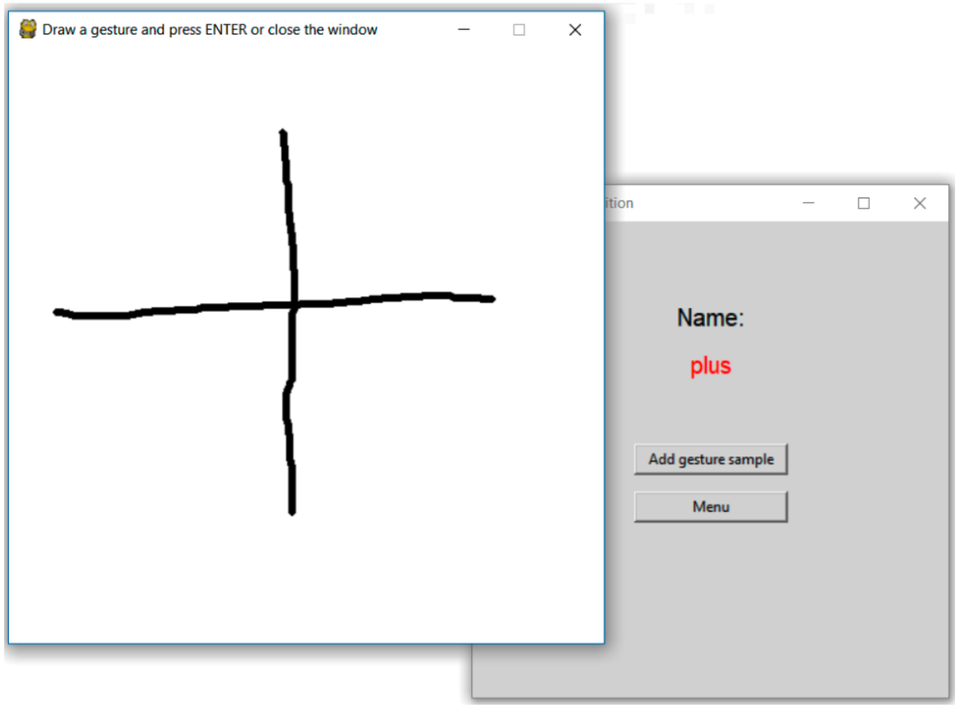


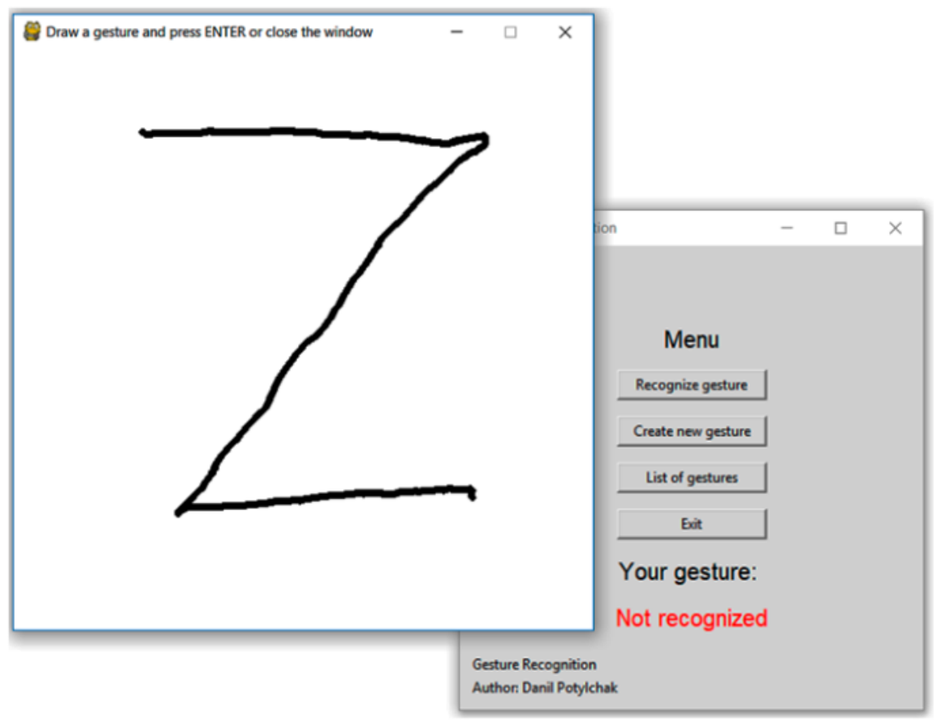


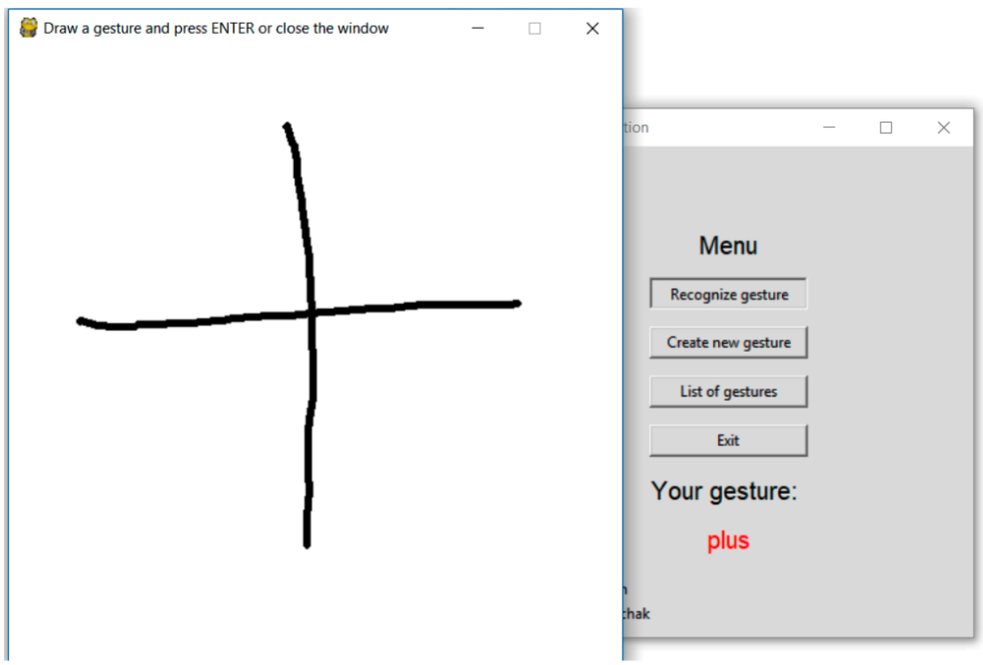




					ДП ІС-5120.1181-с.КЕ			
Зм.	Арк.	№ документа	Підпис	Дата		Літера	Маса	Масштаб
Розробив		Потильчак Д.Г.			*Креслення вигляду екранних форм*			
Перевірив		Баклан І.В.						
						Аркуш 1		Аркушів 2
Н. кон.		Москаленко Н.В			*Інформаційна система підтримки розпізнавання комп'ютерних гестур*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51*		
Затвердив		Баклан І.В.						





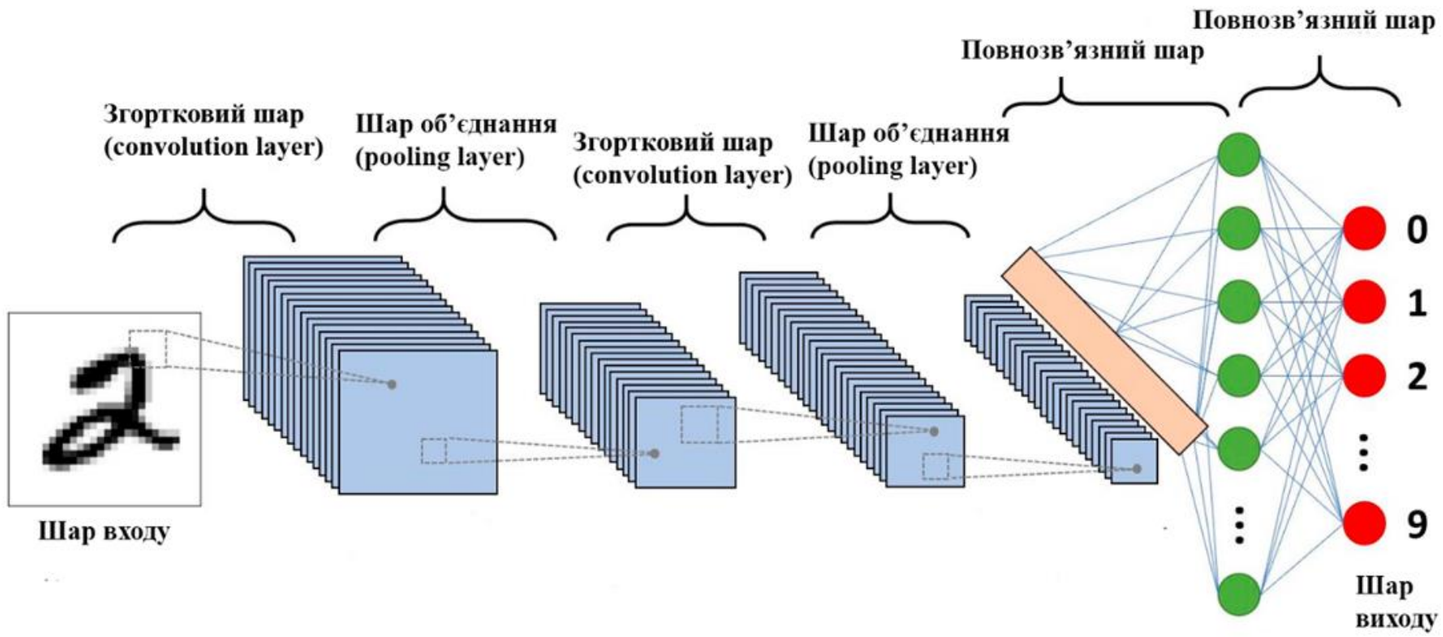


					ДП ІС-5120.1181-с.КЕ		
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм	Літера	Маса / Масштаб
Розробив		Потильчак Д.Г.				Аркуш 2	Аркушів 2
Перевірив		Баклан І.В.					
Н. кон.		Москаленко Н.В			Інформаційна система підтримки розпізнавання комп'ютерних гестур	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51	
Затвердив		Баклан І.В.					

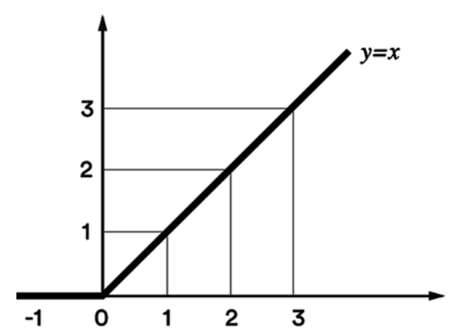
Рішення з математичного забезпечення

Основним завданням є знаходження вагів згорткової нейронної мережі, з якими мережа з достатньою точністю розпізнає гестури

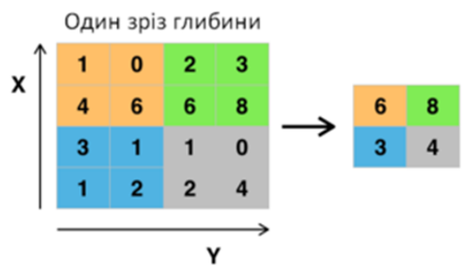
Архітектура згорткової нейронної мережі



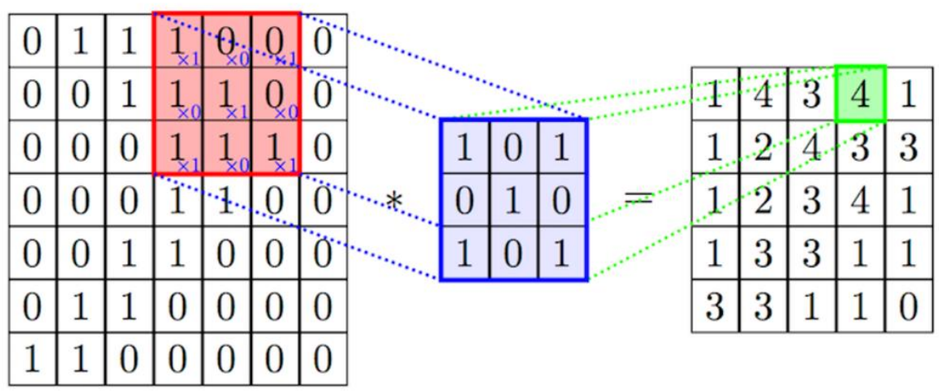
Функція активації RELU



Приклад обчислення агрегаційного шару



Приклад обчислення згорткового шару



Демонстраційний плакат до дипломного проекту

„Інформаційна система підтримки розпізнавання комп'ютерних гестур"

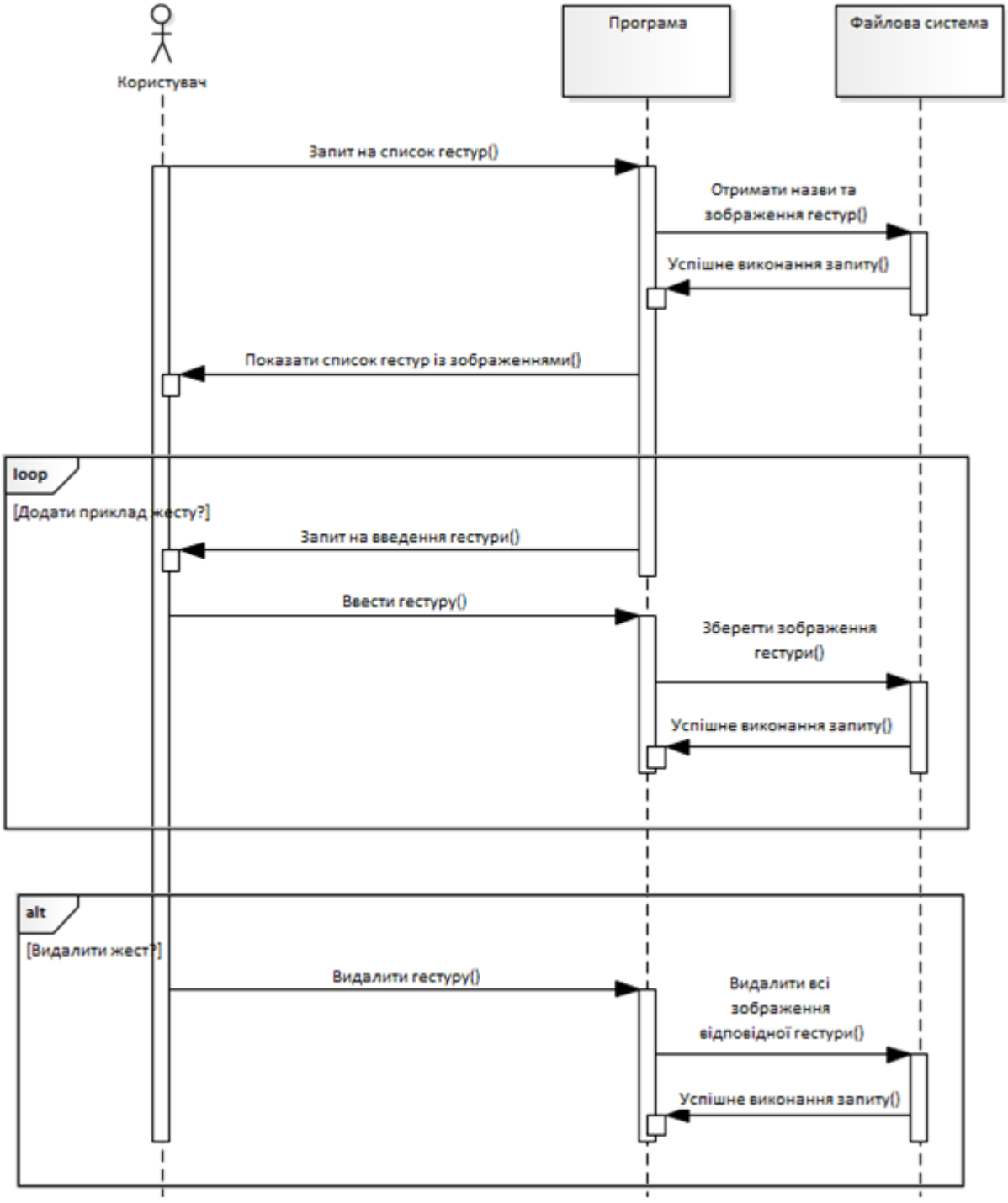
Виконав студент гр. ІС-51 Потильчак Д.Г.

Керівник ДП Баклан І.В.

Діаграма послідовності для процесу створення гестур

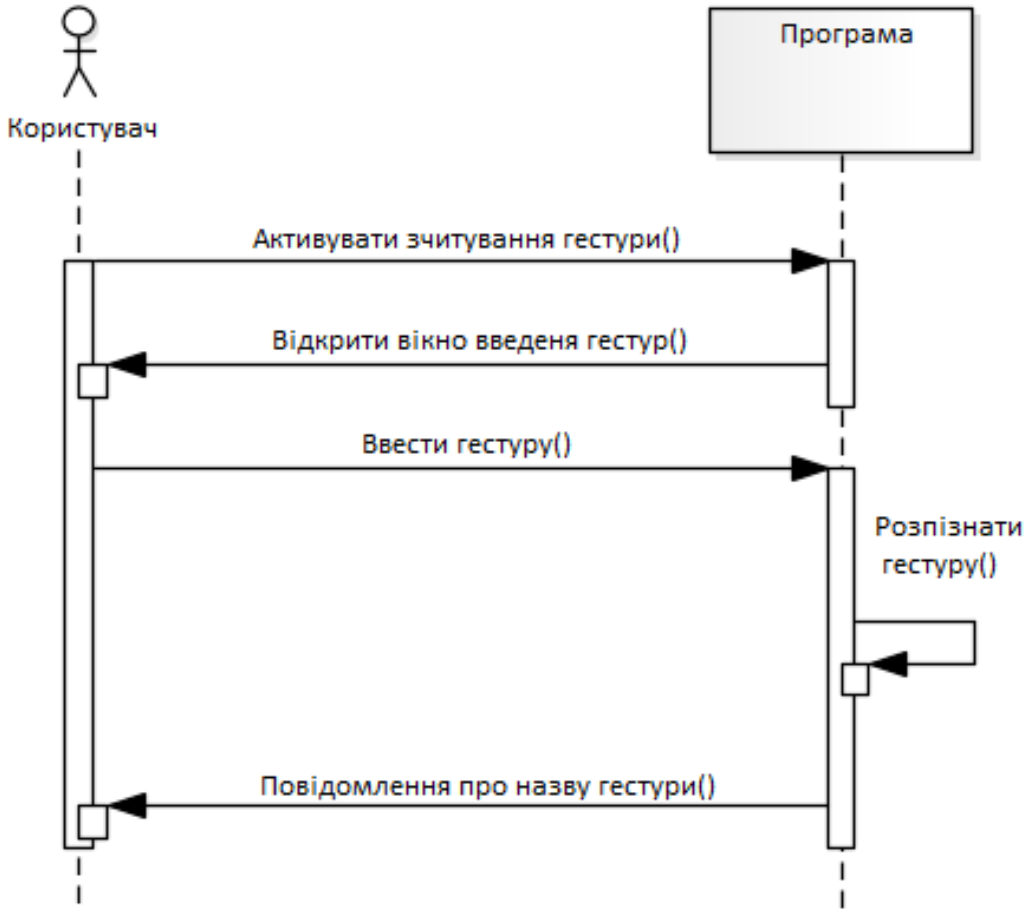
					ДП ІС-5120.1181-с.ССП		
						Літера	Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна послідовності		
Розробив		Потильчак Д.Г.					
						Аркуш 1	Аркушів 3
Перевірив		Баклан І.В.					
Т. кон.					Інформаційна система підтримки розпізнавання комп'ютерних гестур	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51	
Н. кон.		Москаленко Н.В.					
Затвердив		Баклан І.В.					

Діаграма послідовності для процесу редагування гестур
(додавання нових прикладів та видалення гестур)



					ДП ІС-5120.1181-с.ССП		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна послідовності	Літера	Маса / Масштаб
Розробив		Потильчак Д.Г.					
Перевірив		Баклан І.В.				Аркуш 2	Аркушів 3
Т. кон.					Інформаційна система підтримки розпізнавання комп'ютерних гестур	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51	
Н. кон.		Москаленко Н.В.					
Затвердив		Баклан І.В.					

Діаграма послідовності для процесу розпізнавання гестури



					ДП ІС-5120.1181-с.ССП			
						Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна послідовності*			
Розробив		Потильчак Д.Г.						
						Аркуш 3		Аркушів 3
Перевірив		Баклан І.В.						
Т. кон.					*Інформаційна система підтримки розпізнавання комп'ютерних гестур*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51*		
Н. кон.		Москаленко Н.В.						
Затвердив		Баклан І.В.						

					ДП ІС-5120.1181-с.ССР		
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна розгортання*	Літера	Маса / Масштаб
Розробив		Потильчак Д.Г.					
Перевірив		Баклан І.В.				Аркуш 1	Аркушів 1
Т. кон.					*Інформаційна система підтримки розпізнавання комп'ютерних гестур*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51*	
Н. кон.		Москаленко Н.В.					
Затвердив		Баклан І.В.					

					ДП ІС-5120.1181-с.ССВ		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використання	Літера	Маса / Масштаб
Розробив		Потильчак Д.Г.					
Перевірив		Баклан І.В.				Аркуш 1	Аркушів 1
Т. кон.					Інформаційна система підтримки розпізнавання комп'ютерних гестур	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51	
Н. кон.		Москаленко Н.В.					
Затвердив		Баклан І.В.					